EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

Data Management Division

DFS Software

NG/AMS

Next Generation Archive Management System

User's Manual

Doc. No.: VLT-MAN-ESO-19400-2739

Issue: 2

Date: 08/01/2003

DRAFT - FOR INTERNAL USE ONLY

	Name	Date	Signature
Prepared:	J.Knudstrup	08/01/2003	•
Approved:	A.Wicenec	/ /2003	•
Released:	M.Peron/P.Quinn	/ /2003	•

CHANGE RECORD

ISSUE	DATE	SECTION/PAGE AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
1/Preperation 1	29.01.2002	All	First issue.
1/Preparation 2	05.03.2002	All	Added comment after internal review.
1/Preparation 1	08.01.2003	All	Major update with new features.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 About this Guide

1.1 Purpose & Scope

This document is the user's manual for the Next Generation Archive Management System (NG/AMS). NG/AMS is the SW for the Next Generation Archive System [2]. It is in charge of the handling of storage media and of archiving and retrieving data files to/from an NGAS Archive. Numerous other services are provided for carrying out the daily operation of an NGAS Archive System.

This manual contains the information needed for configuring and operating NG/AMS. It is also described how to enhance the system with new features by adding various types of plug-ins. These plug-ins are small Python functions with a specific interface and a specific set of tasks.

The audience of this document is high-level users who wish to perform archiving and retrieval of data files into/from NGAS. However, also more advanced users who need to tune and adapt the system by changing the configuration will find the necessary information in this document. Finally support for the very advanced user is provided. The latter type of user is the user who adds or changes functionality of the system by providing new plug-ins or changing existing ones.

1.2 How to Read this Manual

The intention of this manual is not to provide a 'book' that can be read sequentially chapter by chapter. For the user unknown to NG/AMS it is recommended to read this chapter and Chapter 2 to get an overview of the manual and of NG/AMS and its features. For more specific issues it is suggested to check the index or the table of contents and read the referenced sections in connection with these issues.

The following conventions are used in this manual:

Item	*Description*
<...>	A name in brackets indicates a substitution of the brackets + the name with the contents of the object referred by the name.
`<text>`	Courier font for examples of source code files and configuration files. In addition this font is used for commands as they must be types on the shell.
"<name>"	Names of SW modules, classes, methods, functions, files etc., are contained in quotes.
<element>[.<element>]	Reference to an XML element.
<element>[.<element>]:<attribute>	Used to refer to a specific attribute in an XML document, e.g.: "NgamsCfg.Ngams:CentralUnit".
CFG: <configuration component>	Reference to an element/attribute in the NG/AMS Configuration. For detailed information about the NG/AMS Configuration, consult Chapter 6.
DB: <DB column>	Refers to a DB column. The reference may also be given as: "[<db>.][<table>.]<column".
FILE: <filename>	Reference to a file (within the NG/AMS SW Package).
FUNCTION:/METHOD: <name>	Reference to a specified function or method provided within the NG/AMS Package.

Table 1: Conventions/styles used in the NG/AMS User's Manual.

Some sections are dedicated to the more advanced users of NG/AMS. These sections are marked with "***EXPERT:***". A 'normal user' may want to skip these sections.

The last chapter (27) contains a quick reference to the commands supported by NG/AMS.

1.3 How to Get Help or Report Problems with NG/AMS or this Manual

In case problems are encountered using NG/AMS, bug/problem reports can be submitted via email to:

ngast@eso.org

This also goes for questions and other assistance needed in connection with the usage and enhancement of the system

1.4 Disclaimer

Although great efforts have been invested in designing robust interfaces for the NG/AMS SW e.g. when it comes to the HTTP communication protocol, various XML document formats, and the interfaces of the APIs provided, it should be mentioned that NG/AMS is still in a relatively early phase, and minor changes may have to be introduced in the various interfaces. It will however be attempted to limit the amount of such changes to an absolute minimum. See also Chapter 26, NG/AMS License Conditions.

In general it is attempted to keep NG/AMS compatible with previous versions. However, in cases where it is necessary to change interfaces etc. to comply with standards, it may be that some backwards incompatibilities are introduced.

1.5 Reference Documents

The following documents contain additional information and are referenced in the text:

Reference	*Document Number*	*Issue*	*Date*	*Title*
VLT-MAN-ESO-19400-2739	VLT-SPE-ESO-19400-2534	1	22.06.2001	"DFS Software, Next Generation/Archive Management System", Design Description, J.Knudstrup.
VLT-MAN-ESO-19400-2739	http://archive.eso.org/NGAST	-	-	Next Generation Archive Systems Technologies, A.Wicenec.

Table 2: Reference documents.

1.6 Acronyms

The following abbreviations and acronyms are used in this document:

DB	Database
DAPI	Data Handling Plug-In
DPPI	Data Processing Plug-In
DTD	Document Type Definition
HDD	Hard Disk Drive
HW	Hardware
N/A	Not Applicable
NGAS	Next Generation Archive System
NGAS DB	NGAS (Data Holding) DB
NGAST	Next Generation Archive System Technologies
NG/AMS	NGAS Archive Management System
OS	Operating System
SW	Software
XML	Extensible Markup Language

Table 3: Acronyms used in the NG/AMS User's Manual.

1.7 Glossary

The following glossary is used in this document:

Archive Facility (Site), Archive Cluster	Refers to an NGAS based archive system within an organization, where all data produced are being managed and from where online access to the data and processing facilities are provided. There might be several such Archive Facilities within an organization.
Archive Request	Request from a client of the NG/AMS Server to have a file archived.
Back-Log Buffering	Back-Log Buffering can be carried out by the NG/AMS Server if an error occurs, which makes it impossible to archive the file at that moment. The file will thus be stored temporarily in the Back-Log Buffer Area. The NG/AMS Server (Janitor Thread) will attempt at a later stage to handle the file.
Bad File	A Bad File is a file that could not be accepted for archiving by NG/AMS. I.e. it was rejected by the DAPI handling this file type. This could e.g. be due to a wrong expected size of a FITS file.
Bad Files Directory, Bad Files Area	Area on the disk where files, which are mal-formed are stored. There is a Global Bad File Directory on one of the system disks on each NGAS Node. Apart from that, there is a Bad Files Directory on each archive disk installed.
Data Provider	An NG/AMS Server to which one or more Subscribers have submitted a request for data. Each time a new file becomes available on this NG/AMS system, the Data Provider will check if it should be delivered to one or more of its Subscribers.
Data Subscriber	Client that has subscribed itself to receive a certain kind of data from an NG/AMS Server. In order for a Subscriber to subscribe itself it must send the SUBSCRIBE command. In order to un-subscribe, it must send the UNSUBSCRIBE command.
Disk, Hard Disk Drive, HDD	In the context of NG/AMS the term disk refers to a random access storage device, which can be mounted under UNIX and which has a file system created on it.
Disk Dictionary	A dictionary containing information for the Storage Media available in an NGAS Host.
Disk Set	A consisting of one or two Storage Media; see also Storage Set.
Dynamic Disk Set	An association between two non-completed Storage Media. This association exists as long as NG/AMS is Online. I.e., there is no static link between NGAS Storage Media.
Logical Name	A 'human' readable name that is used when referring to disks. The disks will typically be labeled with the Logical Name. Should be unique, although this may not be guaranteed as for the Disk ID.
Main (Data) File	The copy of the data file stored on the Main Storage Area.
Main Disk	Data is archived onto Storage Sets. The Main Disk is the primary Storage Media of the set and must also be present.
Main (Storage) Area	The array of HDDs in an NGAS Node, which, when filled with data are send to the Archive Facility Site.
NG/AMS Server	The central process of NGAS. It receives the data file from the Data

	Providers, invokes the appropriate DAPI to handle the data and ingests the information about the data in the NGAS DB.
NGAS Cluster	Refers to a set of NGAS Hosts, which are used together as an archive unit.
NGAS Configuration	The XML based document used as input to the NG/AMS Server when this is started. Contains all the information needed for NG/AMS to operate in a given context.
NGAS Host/NGAS Node	Refers to a WS which has the NG/AMS SW installed and which is used as an archiving or data server unit.
Physical Disk Dictionary	Dictionary that contains information about each NGAS Storage Media available at a certain NGAS Node.
Processing Area	Directory used to store temporary copies of files to be processed and other temporary files created during processing.
Production Site	Refers to the site where data is produced and archived, in this case, into the NGAS System.
Production Site, Data Production Site	The site, e.g. at the telescope site, where data is produced and being archived into an NGAS System.
Replication (Data) File	The copy of a data file, which is stored in the Replication Area.
Replication (Storage) Area	The array of HDDs that contains the replicas of the data on the disks in the Main Storage Area.
Staging Area	A storage location (directory) used to temporarily store data files being handled. NG/AMS e.g., uses a Staging Area on each Target Disk, for receiving data files before moving the files to their final location.
Storage Media	Refers to a storage unit used in the context of NGAS for receiving data being archived and from which archived data is retrieved. Used interchangeably with Storage Disk (disk), as HDDs are the media used presently to store data.
Storage Set	A storage unit, which consists of either one or two disks on which data is archived.
Subscriber	See Data Subscriber.
Subscription Back-Log	A buffer in which data files that could not be delivered to Subscribers are stored until a delivery can be carried out successfully.
Target Disk/Target Storage Media	Media selected to receive a file being archived.

Table 4: Glossary used in the NG/AMS User's Manual.

2 Overview

In this chapter the basic concepts of NGAS and NG/AMS are described. An overview of the NG/AMS is given as well as a description of the various fundamental features and services provided by NG/AMS. This chapter provides a somewhat high-level description of the most important features and services. More in-depths descriptions can be found in the subsequent chapters.

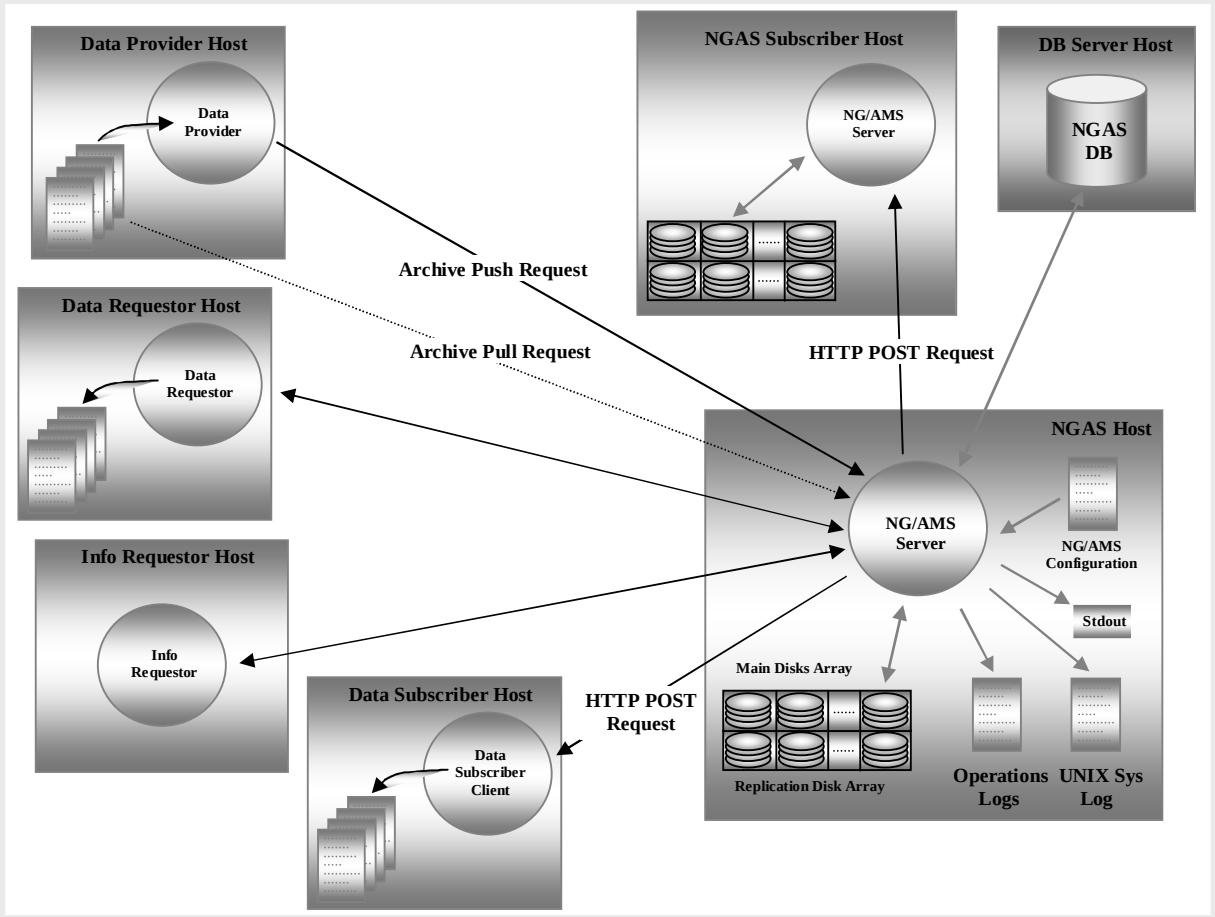

Figure 1: Example operational environment of the NG/AMS Server.

2.1 The Concept of NGAS & NG/AMS

The concept of NGAS is to use random access Storage Media to obtain high I/O performance. However, any device that can be mounted under UNIX (LINUX) and on which a file system can be created, can become an NGAS Storage Media. Storing data on HDDs has several advantages over the present scheme used e.g. by ESO, whereby CD-ROMs and DVDs are used to store the data. Some advantages are:

- The archiving of data files can be carried out very fast.
- Data is online as soon as it has been archived.
- It is not necessary to store data in an intermediate location and to generate later the final media. The final media is create 'real-time'.
- The processing power of the computers hosting the disks can be used to process the data both during archiving and while retrieving data.
- In general an archive system based on NGAS requires much less manual intervention than existing systems based on CD-ROM/DVD disks or tapes.
- The price per storage unit is relatively low compared to other solutions.

The NGAS is based on standard PC HW running Linux. They are normally equipped with a set of HDD sliders in which HDDs can be inserted and removed easily. It is foreseen to have one or more NGAS Hosts at the telescope sites, one in

connection with each major producer of data. As soon as a disk is full, it will be send to the Archive Facility Site where it will be installed in a free slot in an NGAS Host in the archive NGAS Cluster. The data is immediately online as soon as the NG/AMS has 'recognized' the disk. NG/AMS can produce a Replication Disk so that a back-up of the data is available. For more information about NGAS check out **[2]** and the links found at this site. Although the system has been developed on UNIX it may be possible to port the SW relatively easily to other platforms supporting Python like e.g. MS-Windows or MAC-OS.

The philosophy behind the NG/AMS SW is to provide an open architecture that can be extended and adapted to be used as a generic archive facility in many different contexts. Therefore, the NG/AMS SW in itself does not have any specific functionality built-in to handle specific types of data or specific HW. All this 'knowledge' must be implemented and made available for an NG/AMS Server in order to make it carry out the requested tasks. This is done by providing a lot of specific services in the form of plug-ins, which are simple Python functions. Due to this scheme, it should be possible to adapt NG/AMS with a minimum amount of effort to handle e.g. many different kind of data.

The heart of the NG/AMS is the NG/AMS Server. This is a multithreaded server based on the standard HTTP protocol. It can be seen as a dedicated WEB server. Since the server is multithreaded it is possible to issue several requests simultaneously. A number of commands are provided by NG/AMS. For more detailed information about these commands consult Chapter 25. For more information about the technical details of the command interface, consult Chapter 7.

2.2 Services & Features

Some of the main services and features provided by the NG/AMS SW are:

- **Multithreaded Server:** The NG/AMS Server is using threads when handling requests from clients. This means that it is capable of handling several requests simultaneously.
- **HTTP Protocol:** The communication interface of NG/AMS is based on the standard HTTP protocol. This makes it easy to access the server from various clients. It is even possible to interact with the server using a WEB browser (see Section 3.8).
- **Flexible Adaptation via Configuration File:** The NG/AMS Server is configuring itself at start-up, based on a large number of configuration parameters defined in the NG/AMS Configuration, which is an XML document. This makes it possible to adapt the server for specific contexts in a flexible way (see Chapter 6).
- **Cluster Mode:** NG/AMS is prepared for operation of a set of NGAS Nodes in a cluster that constitutes an archive data server and processing facility (see Section 4.1).
- **Adding of Specific Behavior Based on Plug-In Concept:** NG/AMS is implemented in a way so that only the kernel/general functionality is implemented (hard-coded) into the server SW. All the context specific features are provided based on a plug-in scheme making it possible to adapt the server in a very flexible way. As an example of this, the specific handling of data during archiving, is done by a plug-in provided for each type of data (see the Chapters 12-21).
- **State Management:** The NG/AMS Server maintains a State/Sub-State scheme to make it possible to restrict the services provided according to the 'condition' of the server (see Section 2.4).
- **XML Information Exchange:** All information sent back from the server (status messages) are based on XML (see e.g. Chapter 22).
- **APIs for C and Python:** APIs for communicating with the server are provided for applications written in C and Python (see the Chapters 9 and 10).
- **Command Line Utilities:** Two command line utilities for communicating with the NG/AMS Server are provided. These are based on the NG/AMS C and Python APIs (see Section 5.2).
- **SW Modularity/Re-usage:** The NG/AMS SW is implemented as a number of classes and library functions, which can be used to build dedicated servers and other applications if needed (see Chapter 23).
- **Data File Archiving via Push/Pull Technique:** Efficient archiving of data files is provided based on an Archive Pull Technique, whereby NG/AMS picks up files given by a URI, and on an Archive Push Technique, where the data provider writes (pushes) the data to the server (see Section 3.1).
- **Canalization of Data Streams:** Via the configuration file it is possible to define how NG/AMS should stream data onto the various Storage Disks available in an NGAS Host (see Section 2.6).
- **Data Replication:** NG/AMS can handle replication of data files if requested. Also the information for such replicated files is updated automatically in the NGAS DB (see the Section 3.1 and Chapter 15).
- **Logging:** A quite substantial set of information can be logged according to different levels: 1) On stdout, 2) In the UNIX syslog, and 3) In a log file (see Section 3.3).
- **Data Consistency Checking:** If enabled, an NG/AMS Server will run a periodic data consistency check of the data stored on the disks under its control. Via a number of parameters it is possible to adjust quite accurately how much load and how long time this task should take up (see Section 3.9).
- **Production of Disk Labels:** NG/AMS can produce labels for the disk cases on request. The actual SW to operate the printer must be provided in the form of a plug-in (see Section 3.10).
- **Email Notification Service:** A service is provided for notifying subscribers about various events occurring during operation. Examples of such events are errors, disk change requests and data inconsistency reports (see the Sections 3.4 and 3.5).

- **Information Query:** A set of various types of information can be queried via the STATUS command. This information is such as the state of the system, or information about files and disks (see Section 27.12).
- **Data File Retrieval & Processing:** NG/AMS provides a scheme for transparent access to the data. Based on the information in the NGAS DB, a contacted NG/AMS Server can locate the data requested by the user and provide this to the user by acting as a proxy (transparent data access). It can also send back HTTP redirection messages to indicate to the data requestor where to find the data. The C and Python APIs handle the data access completely transparent for the client (see Section 3.2).
- **Access/Service Restriction:** It is possible to enable/disable some basic services via the configuration. The services in questions are for the moment: 1) Handling of Archive Requests, 2) Handling of Retrieve Requests, 3) Data Processing and 4) Remove Requests (removing of disk and file information from the system) (see Section 3.11).
- **Back-Log Buffering of Data:** In case problems occur preventing NG/AMS from archiving data, NG/AMS will Back-Log Buffer data and try to handle this at a later stage (see Section 3.7).
- **Disk Registration & Supervision:** When a disk first has been registered by NG/AMS, the movements of the disk will be monitored by NG/AMS, so that when it appears in an NGAS Host the NGAS DB will be updated to indicate the latest status of the disk (see Section 2.7).
- **File Registration:** A number of parameters are registered for the files archived in the NGAS DB, making it possible to locate, retrieve and process these files (see the Sections 8.3 and 15.1).
- **File Cloning:** A service is provided with which it is possible to clone single files, sets of files, or entire disks (see Section 27.2).
- **Data Subscription Service:** A service is provided to export data being archived to Subscribers interested. A Subscriber can either subscribe to all data being archived on an NGAS Host, or to part of it. Latter is done by means of Filter Plug-Ins that are applied on the data to determine whether to export it or not. In this way, it is e.g. possible to synchronize data holdings between different NGAS Nodes (see Section 4.2).
- **Removing File and Disk Information:** Two commands are provided to remove single files or set of files. Another command is provide to remove an entire disk from the system (see the Sections 27.9 and 27.10).
- **Suspension/Wake-Up Service:** An NG/AMS Server can be configured to suspend the NGAS Host where it is running. A service is provided so that another NGAS Server can be requested to wake up an NG/AMS Server suspending itself (see Section 4.3).
- **Generation of Checksum:** NG/AMS generates a checksum value for each file generated. This is based on a plug-in concept so that context/data specific checksum calculation can be applied (see Chapter 18).
- **Extendable for Usage with Various DBMS':** NG/AMS is prepared for usage with various DBMS'. For now only Sybase is supported, but this can easily be expanded.
- **Simulation Mode:** NG/AMS provides a Simulation Mode, which makes it possible to operate the system without the availability of the actual HW, like the disk controller, disks, etc. Running in Simulation Mode, a simulated NG/AMS environment is generated on a single disk. This is useful for test and development. The Simulation Mode however, could also be used to run an NG/AMS on a 'normal' workstation for archiving data in a production system (see Section 3.6).
- **Thorough Documentation:** Apart from this manual, thorough and accurate documentation contained in the Python source code of NG/AMS is provided. This makes it possible to browse the documentation online e.g. using "`pydoc`"[1] (see Section 23.2).

The services and features listed above and described shortly, are explained in more detail in this and the following chapters.

2.3 Starting & Stopping the NG/AMS Server

The NG/AMS Server can be invoked with a number of different command line parameters. These are described in Section 5.1. It is mandatory to specify an NG/AMS Configuration to be used by the NG/AMS session. How to configure the NG/AMS environment is described in Chapter 6. The server can be started with the "-v" option to produce output on "stdout". Normally, in a production environment, it will be started as a back-ground process, which only produces log output to the UNIX "syslog" and/or a Local Log File (see also Section 3.3).

The server can be stopped either by sending a "SIGTERM" signal (15) or by sending an EXIT command, which can be issued when the server is in Offline State (see also 2.4). If the server is killed with a "SIGTERM" signal, it will invoke internally a signal handler that cleans up the environment and shuts it down in a proper manner whereby also the System Offline Plug-In (Chapter 13) is invoked. Also when issuing an EXIT command, the server invokes the proper 'clean-up procedure'. If the server is killed by a "SIGKILL" (9) signal, the signal handler is not invoked, and the server leaves its environment in an 'undefined' state. This also happens if the computer on which the server is running is shut down abruptly. If this happens it will be necessary to start the server subsequently with the "-force" parameter to force it to start-up. It is possible to 'clean up' the environment by bringing the server Online/Offline in the proper manner.

[1] http://www.python.org/doc/current/lib/module-pydoc.html

2.4 The NG/AMS Server States & Sub-States

The NG/AMS Server is maintaining a scheme of a State and a Sub-State that determine which services the server can handle at a given point in time and which indicate the 'condition' of the server. The States and Sub-States and the corresponding conditions are as follows:

State ⇩ Sub-State ⇨	Idle	Busy
Offline	This is the condition in which the NG/AMS Server enters after starting up, and when the OFFLINE command has been issued. In this state only the STATUS command is accepted. I.e., no Archive or Retrieval Requests are handled. The EXIT command is also accepted. Latter makes the server clean up and terminate.	In this state the server is performing the transition from Offline to Online, or is preparing to exit from execution. No commands are accepted.
Online	In this state the server is ready to handle commands like ARCHIVE and RETRIEVE. In addition the OFFLINE command is accepted.	In this state the server is busy handling one or more Archiving or Data Retrieval Requests. Also the STATUS command is accepted. An OFFLINE command will be rejected.

Table 5: NG/AMS State/Sub-States.

It is possible to query the state of the server by issuing a STATUS command without parameters. The reply to a STATUS command is an XML document with the following contents:

```
<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2002-12-23T13:15:52.194" HostId="acngast1" Message="Successfully handled command STATUS"
          State="ONLINE" Status="SUCCESS" SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
</NgamsStatus>
```

Figure 2: Response to STATUS command.

2.5 The NG/AMS Storage Media Infrastructure

The Storage Media infrastructure used by NG/AMS is depicted in Figure 3.

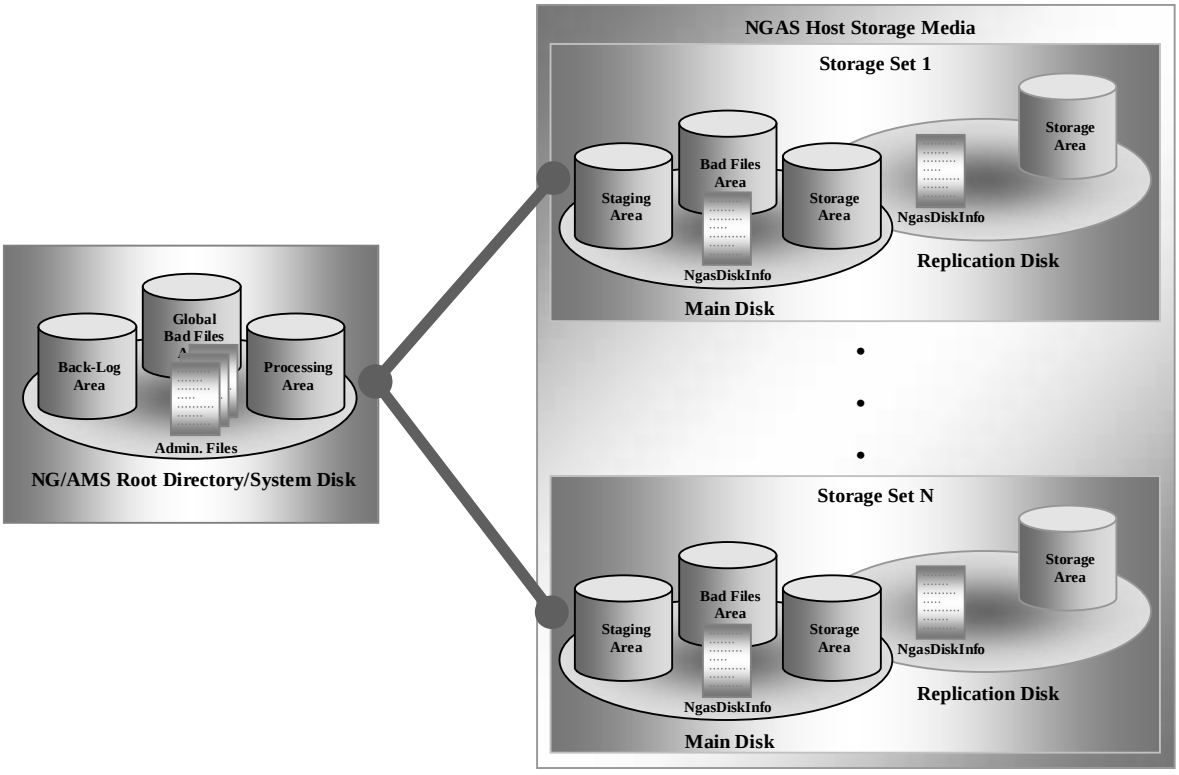

Figure 3 The NG/AMS Storage Media Infrastructure.

During operation (archiving), disks are usually used in pairs as shown in Figure 3. This association of disks however, is handled in a dynamic manner, so that disks are associated with each-other when NG/AMS goes Online and as long as it remains Online. This dynamic association is done, based on the definition of Storage Sets in the NG/AMS Configuration (see Section 6.2, XML element: "StorageSet"). As soon as the server goes Offline, the association does not exist anymore. *This means that in the context of NGAS, data is seen file-wise and not disk-wise and no attempts are made in order to maintain identical sets of files on different disks once associated during operation*. Note, that it is possible to have a Dynamic Disk Set consisting of only a Main Disk. It is recommended however, always to use the Replication Service of NG/AMS to increase data safety, to avoid loss of data.

A Storage Set is considered to be completed when either of the two disks in it are considered as 'full' (see also Section 3.5). If a Storage Set consists only on one Main Disk, it is considered completed when the Main Disk is full. Beware, that since the association between disks only exists during operation, it may be that only one of the disks in a set is actually marked as completed in the DB, whereas the other remains un-completed and can be used together with another disk. In this way it is e.g. possible to use Main Disks of size 80 GB together with Replication Disks of size 200 GB. In the definition of the Storage Set in the NG/AMS Configuration, it is possible to 'lock' two disks together, so that when one of the disks is completed, also the other disk will be marked in the DB as completed (CFG: "NgamsCfg.StorageSet:Synchronize"). Synchronization should normally be used when disks of the same size are used together, to avoid that one disk remains un-completed whereas in reality hardly any space is available on the media.

As seen in Figure 1, the NG/AMS Storage Media infrastructure, is based on a single root directory under which the Storage Disks are mounted. Under this area, NG/AMS is also storing some files for internal purposes. Among these is a file containing the PID of the NG/AMS Server process (FILE: "<NgamsCfg.Ngams:MountRootDirectory>/.NGAS_<NGAS ID>", e.g. "/NGAS/.NGAS-acngast1-7777"). The Back-Log and Global Bad Files Directories can be placed in a location of choice. This is done via the NG/AMS Configuration. The names of these directories are "<NgamsCfg.Ngams:BackLogBufferDirectory>/bad-files" and "<NgamsCfg.Ngams:BackLogBufferDirectory>/../back-log".

The Processing Area (Directory) shown in Figure 3, is used by NG/AMS for storing temporary files while doing file processing. The files stored in this directory will be removed by NG/AMS after the processing has finished. The name of this directory is: "<NgamsCfg.FileHandling:ProcessingDirectory>/processing". Some care should be applied when determining the location of these directories, since it may have an influence on the performance of the system. E.g., if a location for the Processing Area is chosen, which has a poor I/O performance, this may slow down the processing considerably.

It is possible to make NG/AMS carry out replication of the files being archived. This feature can also be disabled (CFG: "<NgamsCfg.Ngams:Replication>"). The data files archived must be stored under a single directory (referred to as Storage Area in Figure 3) in the mount directory on the target disks. The name of this area is configurable (CFG: "<NgamsCfg.FileHandling:PathPrefix>"). It is up to the DAPI implementation to define the structure of the directories and files within the Storage Area. On the data disks there is also a Staging Area used by NG/AMS when receiving data files. Data is received directly onto the Main Target Disk for efficiency reasons. The name of this directory is: "<NgamsCfg.Ngams:MountRootDirectory>/<disk mount directory>/staging". There is only one such Staging Area on the Main Disk. In case a file is identified as bad by the DAPI, it is stored in the Local Bad Files Area. The exact path of this is: "<NgamsCfg.Ngams:MountRootDirectory>/<disk mount directory>/bad-files". A file could be considered as bad e.g. if a checksum value for the file is found to be inconsistent. There is only a Bad Files Area on the Main Disks. Also located on the Data Disks is a file named "NgasDiskInfo". This file is an XML document that contains a summary of the information about the disk contained in the DB. An example of such a file can be found in Section 22.2.

2.6 Data Classification & Handling

One of the fundamental concepts behind NG/AMS is the way data is classified and handled. This is based on the same concept as used by many WEB browsers and mail tools, namely on the mime-type of the data, which again is derived from the extension of the data files. It is also possible to explicitly specify a mime-type for a data file when issuing it for archiving. In NG/AMS no mime-types for the data files handled are hard-coded into the SW. By means of the NG/AMS Configuration, mime-types for new types of data files to be handled can be added. Note that for new types of data a corresponding DAPI must be provided (see Chapter 15). If NG/AMS encounters a data file with an unknown mime-type (not defined in the configuration) while handling an Archive Request, the request will be rejected. Such data files will not be buffered on the NGAS Host handling the request.

It is also possible to define an arbitrary number of Data Streams, normally one per each type of data to be handled. In the data stream the following information must be defined:

Mime-Type	The mime-type indicating for which data the stream is defined.
DAPI + Parameters	The DAPI that should be used to handle the processing and archiving of the data file. In addition parameters for the DAPI can be specified in the configuration file.
Target Storage Set(s)	One or more Storage Sets, on which the data can be stored.

Table 6: Parameters for data classification.

See Chapter 6 for more information about the NG/AMS Configuration.

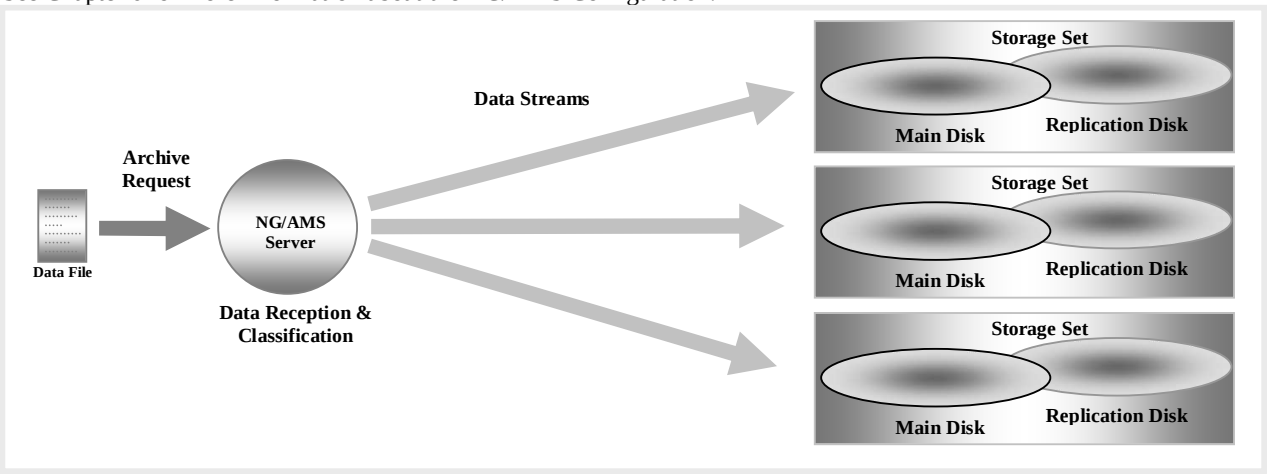

Figure 4: Data channeling.

Two standard mime-type are used by NG/AMS. These are:

text/xml	Used by NG/AMS to indicate that a reply contains an XML document.
ngas/archive-request	Generic mime-type used to indicate for NG/AMS that it should determine the mime-type from the file URI. It is also possible to specify the mime-type explicitly in an Archive Request.

Table 7: Reserved mime-types.

Using the NG/AMS APIs (see the Chapters 9 and 10) the user/client normally does not have to worry about this aspect.

2.7 Disk Handling/Life Cycle of an NGAS Storage Media

In this section the various stages in the life cycle of an 'NGAS disk' are described. In the diagram in Figure 5, a typical life cycle for an NGAS Storage Media is shown. There might be differences for the various contexts to how the actual disk handling is implemented.

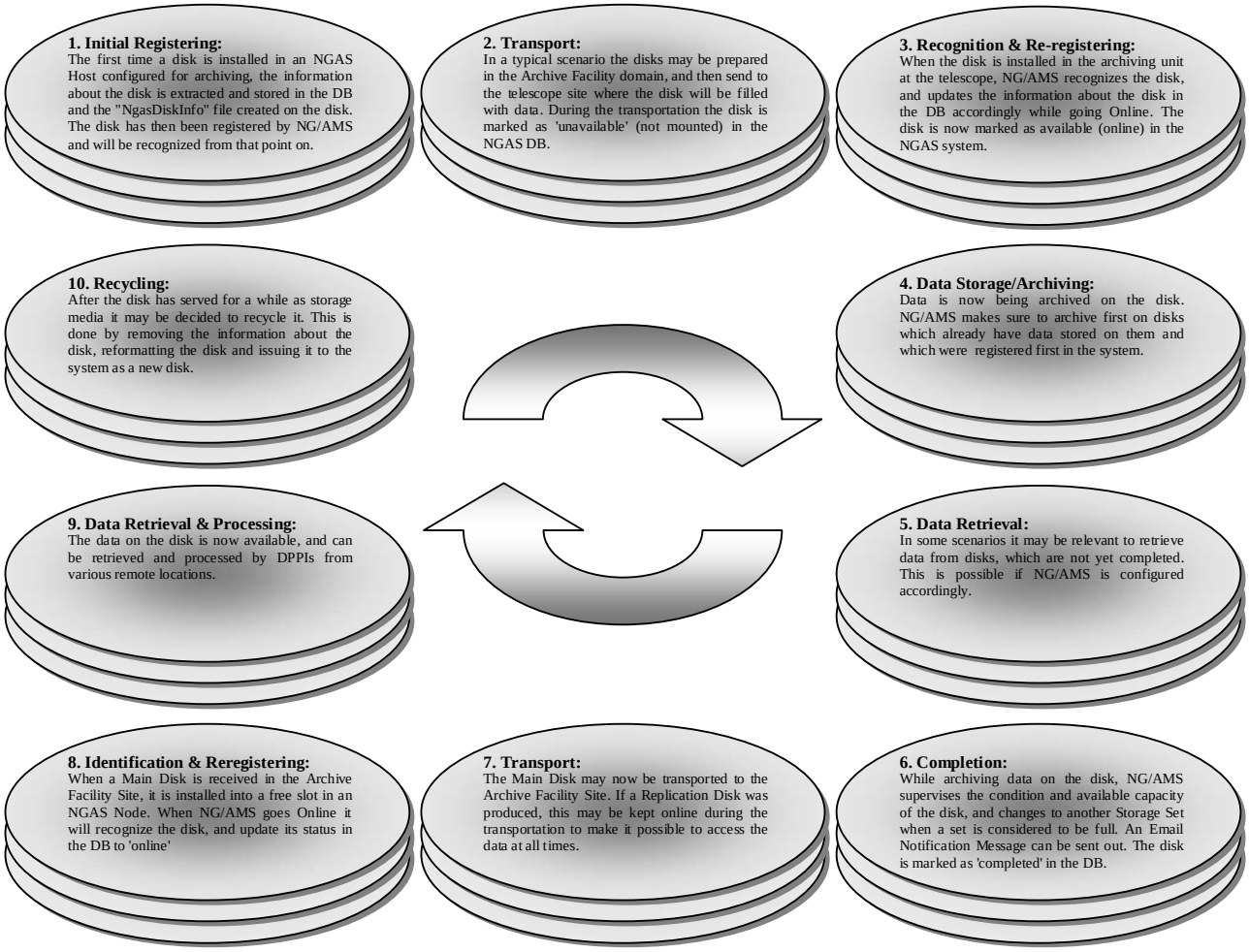

Figure 5: Life cycle of an NGAS disk.

Empty NGAS disks, used for archiving purposes, are usually kept together in pairs. NG/AMS however, does not impose a static association between disks, and the association of non-completed disks (provided by the Logical Name), serves mainly to make the operation of an NGAS based archive system more convenient. For more information about the handling of disk association consult Section 2.5.

3 Basic Features

In this chapter the basic features from the list in Section 2.2 of NG/AMS are described in detail.

3.1 Data File Archiving

Archiving of files in NG/AMS is done via the ARCHIVE command. Data files can be archived either by using the Archive Push Technique or the Archive Pull Technique. Using the former, the application archiving the file, reads in the file and sends the contents of the file in the HTTP message body. When making use of the latter technique, the client sends a URL indicating a location where the file can be picked up by NG/AMS. This URL must therefore be accessible for NG/AMS.

NG/AMS maintains a scheme of file versioning, whereby if a file with the same File ID is issued several times, the version number in the NGAS DB (DB: "ngas_files.file_version") is incremented by one. The first file with a given ID has version number 1.

In short, the handling of an Archive Request is done as follows:

- NG/AMS receives the Archive Push or Pull Request.
- It determines the mime-type of the data file. From the configuration and the current disk status, which it reads from the NGAS DB, it finds a suitable Target Disk for the data.
- The data is subsequently received into the Staging Area of the Main Target Disk.
- Afterwards the DAPI corresponding to the type of data in question, is invoked to do the specific handling of the data.
- After the DAPI has finished its processing, it returns control to NG/AMS.
- The DAPI has then extracted/produced the necessary information for NG/AMS to be able to update its status in the DB and to be able to move the file to its final destination.
- If replication is enabled, NG/AMS also carries out this action and updates the information for the Replication File in the DB.

The ARCHIVE command is described in Section 27.1. A more detailed description of the procedure executed by NG/AMS while handling an Archive Request, can be found in Chapter 15 (describing the DAPI).

3.2 Data File Retrieval & Processing

Archived files can be retrieved from NG/AMS using the RETRIEVE command. In the present implementation it is only possible to retrieve one file at a time. It is possible to request to have the data processed by a DPPI before NG/AMS returns it. The concept of the DPPIs is described in detail in Chapter 17.

When NG/AMS receives a Retrieve Request, it checks the NGAS DB for the various instances of a data file with the given ID, which are online. Several versions may be available. The decision of which file to choose, is done as follows:

- A list of all files with the given File ID, which are registered as being online, is retrieved from the NGAS DB. In addition, files marked to be 'ignored' are not considered.

- The files are ordered according to their File Version, whereby the latest file (highest version number), gets the highest priority.

- It is subsequently investigated where the instances of the given file are located. Three cases are considered: 1) Local Host - The file is stored on the NGAS Host handling the request, 2) Private Network - The file is stored on an NGAS Host within the private network (IP address of the format: 10.X.X.X), 3) Remote Location - The file is stored on a remote NGAS Host not contained within private network. I.e., if a file with a given ID and the same version is found on several NGAS Hosts, the selection criteria for which file to take is done according to the priority list described in this paragraph. I.e., if a file is found on the local host and on a remote host, the instance on the local host is taken.

- When a file is found to be residing on the local host or on an NGAS Host within the private network, NG/AMS always gets the file, processes it if requested and sends it back to the requestor. If the file is located on a remote host, NG/AMS will either send back an HTTP redirection response (HTTP Status Code: 303), or it can act as

proxy. In the former case it is up to the requestor to re-issue the request to the NGAS Host referenced in the redirection HTTP response (see also Section 7.3). In the latter case, it retrieves the file from the remote location, and subsequently sends back this file to the requestor. Whether an NG/AMS Server should act as a proxy or not, is configurable (CFG: "NgamsCfg.Ngams.ForceProxyMode", 1 = Proxy Mode).

If a file is located on a host within the private network or on a remote host, possible processing requested will be carried out on the host on which the file resides.

Methods for retrieving files in an easy manner are provided by the C and Python APIs. See the Chapters 9 and 10.

3.3 Logging

A number of different types of log output can be produced by NG/AMS. These and their properties are:

Log Type	Description
Local Log File	The location of a Local Log File is defined in the NG/AMS Configuration file (CFG: "NgamsCfg.Log:LocalLogFile"). I.e., the user can decide himself where to put this file. The level (intensity) with which there is logged, can be adjusted as well (CFG: "NgamsCfg.Log:LocalLogLevel"). Note, that NG/AMS will continue to write (append logs) in the same log file. I.e., it should be considered to implement means to purge the log file[2] periodically. The format of the Local Log file log entries is as follows: <ISO 8601 time stamp> **[**<type>**]** <log message> **[**<source file>**:**<method>**:**<line**:**<thread>**]**[3] where <type> is defined as: <type> := EMERGENCY | ALERT | CRITICAL | ERROR | WARNING | NOTICE | INFO | DEBUG Examples of some entries in a Local Log File are: … 2002-12-19T17:00:20.640 [INFO] Handling HTTP request: client_address=('134.171.21.30', 34094) - method=POST - path=|ARCHIVE| - content-disposition=attachment; filename="SmallFile.fits";wait="1";no_versioning="0" - content-type=ngas/archive-request - host=acngast1 - content-length=69120 [ngamsServer.py:handleHttpRequest:950:Thread-2] 2002-12-19T17:00:20.650 [INFO] Received command: ARCHIVE [ngamsCmdHandling.py:cmdHandler:889:Thread-2] 2002-12-19T17:00:20.650 [INFO] Handling Archive Push Request ... [ngamsCmdHandling.py:handleCmdArchive:580:Thread-2] 2002-12-19T17:00:20.670 [INFO] Archiving file: SmallFile.fits with mime-type: image/x-fits ... [ngamsArchiveUtils.py:dataHandler:586:Thread-2] 2002-12-19T17:00:20.790 [INFO] Plug-In handling data for file with URI: SmallFile.fits [ngamsFitsPlugIn.py:ngamsFitsPlugIn:129:Thread-2] 2002-12-19T17:00:22.540 [INFO] NGAMS_INFO_FILE_ARCHIVED:4020:INFO: Successfully archived file with URI: SmallFile.fits. Time: 1.874s [ngamsArchiveUtils.py:dataHandler:659:Thread-2] …
(UNIX) Syslog	It is possible to instruct NG/AMS to produce log entries into the UNIX syslog. This is only done when certain important events occur. Such events are error conditions, and handling of archive requests. Whether or not to log into syslog is specified in the configuration file (CFG: "NgamsCfg.Log:SysLog"). It is possible to specify an ID, which is written in each log entry in the syslog (CFG: "NgamsCfg.Log:SysLogPrefix"). This makes it possible to filter out logs for a certain context at a later stage. An example of some syslog entries produced by NG/AMS is: … Feb 20 12:58:04 w2p2nbu python: DFSLog:2002-02-20T12:58:04.200 Error w2p2nbu NGAMS_ER_DISK_INACCESSIBLE:3004:ERROR: Disk with ID: Slot ID: 3 - Disk ID: IC35L080AVVA07-0-VNC400A4C1G8RA is not accessible (writable). Feb 20 12:58:04 w2p2nbu python: DFSLog:2002-02-20T12:58:04.410 Error w2p2nbu NGAMS_ER_DISK_INACCESSIBLE:3004:ERROR: Disk with ID: Slot ID: 4 - Disk ID: IC35L080AVVA07-0-VNC400A4G0KZ8A is not accessible (writable). Feb 21 23:43:56 w2p2nbu python: DFSLog:2002-02-21T23:43:56.800 Notice w2p2nbu Disk with ID: IC35L080AVVA07-0-VNC400A4C1607A - Name: LS-FitsStorage3-M-000027 - Slot No.: 5 - running low in available space (4938 MB)! Feb 22 09:29:40 w2p2nbu python: DFSLog:2002-02-22T09:29:40.740 Notice w2p2nbu Marked Main Disk with ID: IC35L080AVVA07-0-VNC400A4C1607A - Name: LS-FitsStorage3-M-000027 - Slot No.: 5 - as 'completed' - PLEASE CHANGE! Feb 22 09:29:40 w2p2nbu python: DFSLog:2002-02-22T09:29:40.770 Notice w2p2nbu Marked

[2] A mechanism for this may be provided later within NG/AMS.
[3] Characters in bold are part of the contents.

	Replication Disk with ID: IC35L080AVVA07-0-VNC400A4C1622A - Name: LS-FitsStorage3-R-000027 - Slot No.: 6 - as 'completed' - PLEASE CHANGE! …
Verbose Log	The Verbose Logs are written to stdout. They contain more detailed information than the two other types of logs. This type of log is usually used for debugging, trouble shooting and test purposes. The Verbose Log Level is adjusted via a command line parameter (-v <level). If this parameter is not specified, no Verbose Log output is produced. The format of the Verbose Logs is as follows: <ISO 8601 time stamp>:<module>:<method>:<line no.>:<log type>> <log message> The various values for <type> are defined in connection with the Local Log File. An example of Verbose Log output is (lowest Log Level (=1)): … 2002-12-23T14:38:53.380:ngamsServer.py:handleHttpRequest:950:Thread-1:INFO> Handling HTTP request: client_address=('134.171.21.30', 34398) - method=POST - path=|ARCHIVE| - user-agent=NG/AMS C-API - content-disposition=attachment; filename="/home/ngasmgr/tmp/WFI.2001-09-15T22%3A49%3A07.652.fits"; wait="1"; no_versioning="0" - content-type=ngas/archive-request - content-length=141546240 2002-12-23T14:38:53.380:ngamsCmdHandling.py:cmdHandler:893:Thread-1:INFO> Received command: ARCHIVE 2002-12-23T14:38:53.390:ngamsCmdHandling.py:handleCmdArchive:580:Thread-1:INFO> Handling Archive Push Request ... 2002-12-23T14:38:53.390:ngamsArchiveUtils.py:dataHandler:586:Thread-1:INFO> Archiving file: WFI.2001-09-15T22:49:07.652.fits with mime-type: image/x-fits ... 2002-12-23T14:38:59.230:ngamsFitsPlugIn.py:ngamsFitsPlugIn:129:Thread-1:INFO> Plug-In handling data for file with URI: WFI.2001-09-15T22:49:07.652.fits 2002-12-23T14:39:48.410:ngamsArchiveUtils.py:dataHandler:659:Thread-1:INFO> NGAMS_INFO_FILE_ARCHIVED:4020:INFO: Successfully archived file with URI: WFI.2001-09-15T22:49:07.652.fits. Time: 55.018s …

Table 8: The supported log output formats.

The Log Level is a number in the range from 1 to 5, whereby 1 is the 'high-level' logs and 5 is the lowest (deepest) level, providing the most thorough information. The interpretation of the various Log Levels is as follows:

Level	Description
1	The lowest Log Level, which only provides a brief summary of the actions performed. Errors and warnings are always logged.
2	This level provides more thorough information of the actions performed.
3	This level performs a quite extensive set of logs describing in details the various actions carried out by NG/AMS and the plug-ins invoked by this. For logging in the log file, there should normally not be logged with a higher level than 3.
4	This level provides a very profound set of information. It is usually only used for debugging and test purposes and for locating errors occurring in the system.
5	The deepest level provides a quite extensive set of logs. Some of the log will be quite repetitive, and logs may be produced cyclically from e.g. the Janitor and the Data Consistency Check Services. The quantity of log information produced is quite big, and if logging into a log file with this level, care should be taken that it may grow in size very rapidly.

Table 9: Interpretation of Log Levels.

The level (intensity) with which there should be logged as well as name of Local Log File and a prefix for the syslog entries, can be specified in the NG/AMS Configuration (CFG: "NgamsCfg.Log"). For further information about this specific properties see Chapter 6.

3.4 Email Notification

Apart from the various types of logging described in Section 3.3, it is also possible to instruct NG/AMS to notify various recipients about important events occurring via email.

The various types of Notification Message are:

Event	*Description*
Alert Notification	An Alert Message is generated as a result of a serious problem encountered. Such a problem may not be recoverable, and it is likely necessary to do some manual intervention. Normally preventative actions should be undertaken immediately.
Error Notification	An Error is the result of a problem encountered, which is not of a very severe character. Often an error situation is provoked by an external request, which is illegal for some reason.

	Could e.g. be that it is tried to archive a file when the system is in Offline State. Depending on the type of error, intervention should be undertaken (ASAP).
Disk Space Notification	A Disk Space Notification is sent out when a certain threshold of minimum free disk capacity is reached. This message is meant only as a 'warning' indicating that the Storage Set is about to be full. No actions are needed, apart from maybe verifying that Storage Sets with free disk space are available.
Disk Change Notification	A Disk Change Notification is send out to indicate that a Storage Set is full and should be removed from the archiving unit and normally replaced with a new Storage Set. See also Section 3.5.
No Disk Space Notification	If no more free Storage Sets are available, a No Disk Space Notification Message is send out to the subscribers of this event. Since this is a severe problem, a special Notification Message is dedicated for referring to this specific problem.
Data Inconsistency Notification	If the Data Consistency Check Service encounters errors/problems with data files, a Data Error Notification Message is send to the subscribers of this event. The files that were found to be 'bad' in some way should be analyzed to find out what is causing the problem. It could be caused by physical problems of the disk, or that due to long storage on the disk, failures start to occur. Problems with a 'problematic' file are normally only reported once. I.e., if the problem is not solved, there will be no more notification about the problematic file until the NG/AMS Server is re-started.

Table 10: The different types of Notification Messages.

The Notification Setup is configured in the NG/AMS Configuration (CFG: "NgamsCfg.Notification"). For further information about this specific property see Chapter 6. An example Disk Change Notification Message can be found in Section 3.5.

3.5 Disk Space Monitoring

During the archiving process, NG/AMS monitors constantly the state of the set of disks currently installed. If the amount of data on a Storage Set reaches a certain limit defined by a configuration parameter, a Notification Message can be send out to a list of subscribers for this event (see 3.4). This event is a pre-warning that this Storage Set is going to be completed (full) within a limited time. The latter depends on the threshold defined in the configuration file. When a Storage Set is considered as ‘completed’, another type of Notification Message can be broadcast to a number of subscribers. This message will indicate that the Storage Set is full and needs to be replaced. The appearance of such an email message is as follows (example):

```
Subject: NGAS-w2p2nau-7777: CHANGE DISKS
Date:    Fri, 25 Jan 2002 01:06:26 +0100 (MET)
From:    ngasmgr.w2p2nau@eso.org

Notification Message:

PLEASE CHANGE DISKS:

Main Disk:
- Logical Name: FitsStorage-M-000024
- Slot ID:      5

Replication Disk:
- Logical Name: FitsStorage-R-000024
- Slot ID:      6
```

Figure 6: Disk Change Email Notification Message.

The Logical Name(s) (Disk Label(s)) as well as the Slot in which the disk(s) are hosted are indicated in the mail. When such a message is received by the NGAS responsible (operators) it is advisable to carry out the suggested changes as soon as possible to avoid saturation. If only a single disk in a set is completed, the Email Notification will only indicate the name of this completed disk (see the Sections 2.5 and 2.7).

3.6 Simulation Mode

It is possible to operate the NG/AMS Server in Simulation Mode, whereby a number of features are disabled or are working slightly different than in Normal Mode. One of the major differences is that it is possible to run without the availability of ‘real’ storage disks. Simulated storage disks are created as directories in the Mount Root Point. These are of the format: "<mount root point>/<storage set ID>-Main | Rep-<simulation slot ID>".

Another difference compared to running in 'real mode' is that the Online and Offline Plug-Ins are not executed since no disks need to be mounted or unmounted. The disk information about the disks is generated/simulated and written in the DB.

For the clients of NG/AMS there is no visible difference between running in Normal Mode or in Simulation Mode. Also the internal aspects are the same, so that e.g. the DB is updated in the same manner in Simulation Mode as in Normal Mode. The Simulation Mode can be quite useful for developing and testing e.g. DAPIs and DPPIs.

It should be mentioned that it is possible to have a fully operational NG/AMS installation running in Simulation Mode on a 'normal' workstation archiving and retrieving data to/from one of the system disks of the workstation.

To enable/disable the Simulation Mode, the attribute "NgamsCfg.Ngams:Simulation" in the configuration is used. See also Chapter 6.

3.7 Back-Log Buffering

Back-Log Buffering is used to temporarily buffer data, which for some reason, not necessarily related to the quality of the data, prevents NG/AMS from performing a proper archiving of the data file. An example of such an event, is e.g. if the DB connection is lost temporarily.

As shown in Figure 3, the Back-Log Buffer Area could be located in the NG/AMS Root Mount Directory as it is practical to collect the data of NG/AMS under a single point. If a problem occurs during the handling of an Archive Request, a file with a unique name will be created in this area and the data of the request buffered in this file. The reply to the Archive Request will indicate the problem, i.e. that the data was Back-Log Buffered. No further actions are needed from the client that issued the Archive Request. Figure 7 shows an example of a reply from NG/AMS when back-Log Buffering was done.

```
ngasmgr@acngast1:/opsw/NGAS/ngams/ngamsData> ngamsCClient -port 7777 -host acngast1 -status -cmd ARCHIVE -fileUri
~/tmp/SmallFile.fits

<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2003-01-08T16:44:40.562" HostId="acngast1" Message="NGAMS_WA_BUF_DATA:4015:WARNING: Problems
          occurred while handling file with URI: SmallFile.fits. Data will be  buffered, and attempted archived
          at a later stage. Previous error  stack: NGAMS_ER_DB_COM:2002:ERROR: Problems communicating with the
          DB: Error: connection is not open." State="ONLINE" Status="FAILURE" SubState="IDLE"
          Version="v2.0-Beta2/2002-12-04T09:22:53"/>
</NgamsStatus>
```

Figure 7: Example reply when Back-Log Buffering is applied.

The NG/AMS Server has an internal thread, Janitor Thread, which runs periodically and tries to clean up the NG/AMS environment. One of the tasks performed is to archive Back-Log Buffered data. If such an attempt fails due to one of the reasons justifying for Back-Log Buffering, the data will be kept in the Back-Log Buffer and a new attempt to archive it repeated later. If the attempt fails for another reason, the data will be moved to the Global Bad Files Area shown in Figure 3. In this case a Notification Message will be sent out to the subscribers of Error Notification Messages, and the appropriate information logged in the log output targets specified (see the Sections 3.3 and 3.4).

In the NG/AMS Configuration it can be specified if Back-Log Buffering should be performed, as well as the parent directory for the Back-Log Buffer Directory. For further information about this specific property consult Chapter 6.

3.8 The NG/AMS Server Command Interface

The NG/AMS Server command interface is based on the standard HTTP protocol. This makes it possible to interface to the NG/AMS Server from different kinds of clients in a simple and straightforward manner. E.g. from a WEB browser (better if XML enabled) it is possible to query the status of an NG/AMS Server:

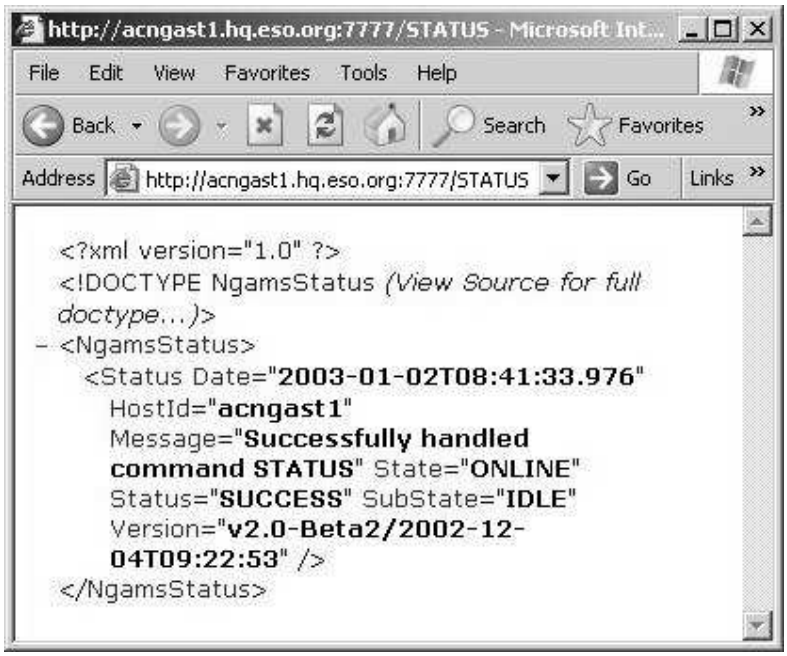

Figure 8: Interaction with an NG/AMS Server from a WEB browser.

Also a standard utility like "telnet" can be used to interact with NG/AMS, e.g. to issue a command like OFFLINE:

```
ngasmgr@acngast1:/opsw/NGAS/ngams> telnet acngast1 7777
Trying 134.171.21.30...
Connected to acngast1.
Escape character is '^]'.
GET STATUS

HTTP/1.0 200 OK

<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2002-12-23T14:59:42.724" HostId="acngast1" Message="Successfully handled command STATUS"
          State="ONLINE" Status="SUCCESS" SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
</NgamsStatus>
Connection closed by foreign host.
ngasmgr@acngast1:/opsw/NGAS/ngams>
```

Figure 9: Example of interaction with NG/AMS using "telnet".

In general, the NG/AMS Python or C based command interface tools, should be used when interacting with NG/AMS from the shell. See Section 5.2 for more information about these tools.

For more in-depth information about the NG/AMS command interface, consult the Chapters 27 and 7.

3.9 Data Consistency Checking

The NG/AMS Server can be configured to carry out a periodic consistency check of the data files, which are stored on the disks installed on that NGAS Node. The following checks are carried out:

- It is checked if files are registered in the DB but are not found on the disk.
- Checksum value for each file is checked according to the value registered in the NGAS DB for the file.
- It is checked if files are found in the Storage Area of the storage disks, which are not registered in the DB.

In case discrepancies are found in the data holding on the disks in connection with an NGAS Host, a Data Inconsistency Notification Message is send out. This has the format, e.g.:

```
Subject: NGAS-arcus2-7778: DATA INCONSISTENCY(IES) FOUND
Date:    Fri, 25 Jan 2002 01:06:26 +0100 (MET)
From:    jknudstr@eso.org

Error Message:

DATA INCONSISTENY(IES) FOUND IN DATA HOLDING:
Date:            2002-02-12T15:32:05.424
NGAS Host:       arcus2
Inconsistencies: 1

Problem Description                    File ID                          Version
-------------------------------------------------------------------------------
ERROR: Inconsistent checksum found     TEST.2001-05-08T15:25:00.123     3
-------------------------------------------------------------------------------
```

Figure 10: Example of a Data Consistency Checking Report.

If files are found, which do not have the checksum properly set, NG/AMS will calculate the checksum using the DCPI specified in the configuration, and send a Data Inconsistency Notification Message to the subscribers of this type of message.

It is possible to enable and disable the Data Consistency Checking Service (CFG: "NgamsCfg.-FileHandling:DataCheckActive"). In addition it is possible to allocate a priority to the data checking thread to calibrate the CPU consumption (CFG: "NgamsCfg.FileHandling:DataCheckPrio"). It is also possible to specify how disks and files are checked, whereby this can either be done sequentially or randomly (CFG: "NgamsCfg.FileHandling:DataCheckDiskSeq", "NgamsCfg.FileHandling:DataCheckFileSeq"). A minimum cycle time for one iteration of the service can also be defined (CFG: "NgamsCfg.FileHandling:DataCheckMinCycle"). If the checking is carried out in less then the specified minimum cycle time, the service will be suspended for a while. A parameter is used to configure the service to produce a log entry after each iteration with summary information about the check carried out. This log entry has the following contents (example log entry taken from the Local Log File):

```
2002-02-26T02:52:00.640 [INFO] Number of files checked: 9529. Amount of data checked: 582478.078 MB. Time for
checking: 25139.280 s
```

Figure 11: Example of a Data Consistency Checking Status Log.

The configuration parameters mentioned above are described in more detail in Chapter 6.

3.10 Label Printing

A label to stick on the disk cases can be produced by NG/AMS by means of the LABEL command. The text on the label is the Logical Name allocated to the disk. In addition printed on the label is the host ID and the Slot ID. An example of a label is as follows (generated by the Brother P-Touch 9200 DX label printer):

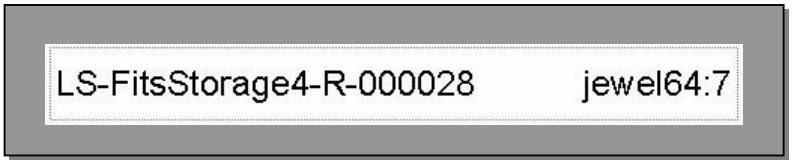

Figure 12: Example Disk Label as generated by NG/AMS.

The part with the Host ID + Slot ID should be removed from the label before sticking it on to the disk case.

The LABEL command takes as input the Slot ID for the disk in question. In addition the Host ID of the host in which the disk is installed.

The label is printed by the Label Printer Plug-In (see Chapter 14).

3.11 Security

It is important to keep in mind that the NG/AMS SW does not come with any security mechanisms built-in when it comes to preventing undesirable intruders (hackers) from connecting to the server and invoking services. This must all be handled

at the level of the operating system and network (firewalls etc.). What is supported are checks to disable certain services and to ensure that only a limited set of plug-ins can be invoked by clients.

The high level NG/AMS services that can be enabled/disabled via the NG/AMS Configuration are:

Service	*Description*
Archive Request Handling	It is possible to enable/disable acceptance of Archive Requests. This is e.g. relevant in an archive data server cluster configuration where no files are being archived. Note that if archiving is disabled, apart from the ARCHIVE Command also the CLONE Command will not be accepted. In addition it will no be possible for NG/AMS to handled back-logged buffered data nor will it be possible to act as Data Subscriber (CFG: "NgamsCfg.Ngams.AllowArchiveReq").
Retrieve Request Handling	Disabling of handling of Retrieve Requests, may be applied for NGAS Node used as archiving units, where it is not desirable that handling of external data retrieval disturbs/loads the system (CFG: "NgamsCfg.Ngams:AllowRetrieveReq").
Processing Request Handling	In connection with a Retrieve Request it is possible to specify that data processing should be applied on the data before replying to the requestor. This may be relevant to avoid to load an NGAS Host too much if handling of the Retrieve Requests themselves is high-priority and where processing would load the system too much to get access to the data within a limited period of time (CFG: "NgamsCfg.Ngams.AllowProcessingReq").
Remove Request Handling	If this feature is disabled, no REMFILE and REMDISK commands will be accepted by the NGAS Host, and it is thus not possible to delete any information in the NGAS system. This should usually be applied e.g. for NGAS Nodes operating in an NGAS data server cluster (CFG: "NgamsCfg.Ngams.AllowRemoveReq").

Table 11: NG/AMS High Level Services that can be enabled/disabled.

See also Section 6.2/"Ngams" Element.

Apart from disabling handling of REMFILE/REMDISK commands (Table 11), it might be advisable to implement additional schemes for preventing data from being deleted from an archive data server NGAS system. This could e.g. be done within the System Online Plug-In (see Chapter 12). One of the responsibilities of this plug-in is to mount the Storage Media available in the NGAS Node. In case a Storage Media is marked as 'completed' in the NGAS DB, the media could be mounted read-only to prevent data from being (accidentally) removed. NG/AMS does not provide any features to prevent data from being as such, and it is up to the designers of the NGAS environment to define how to provide this.

4 *EXPERT:* Advanced Features

In this chapter the advanced features of NG/AMS listed in Section 2.2 are described in detail.

4.1 *EXPERT:* Operation in Cluster Mode

For larger data holdings, it will normally be necessary to have a maybe large number of NGAS Nodes to provide online access to the data in an Archive Facility. It is therefore important to make a proper design of the architecture of such a Archive Facility Cluster.

NG/AMS provides a few services to support such operation in 'cluster mode'. Among this is the capability of NG/AMS to act a proxy while handling a Retrieve Request (see also Section 3.2). In addition NG/AMS distinguishes between NGAS Nodes globally accessible, and nodes within a private network. With this simple scheme, it is possible to build up e.g. a hierarchical cluster as the one shown in Figure 13.

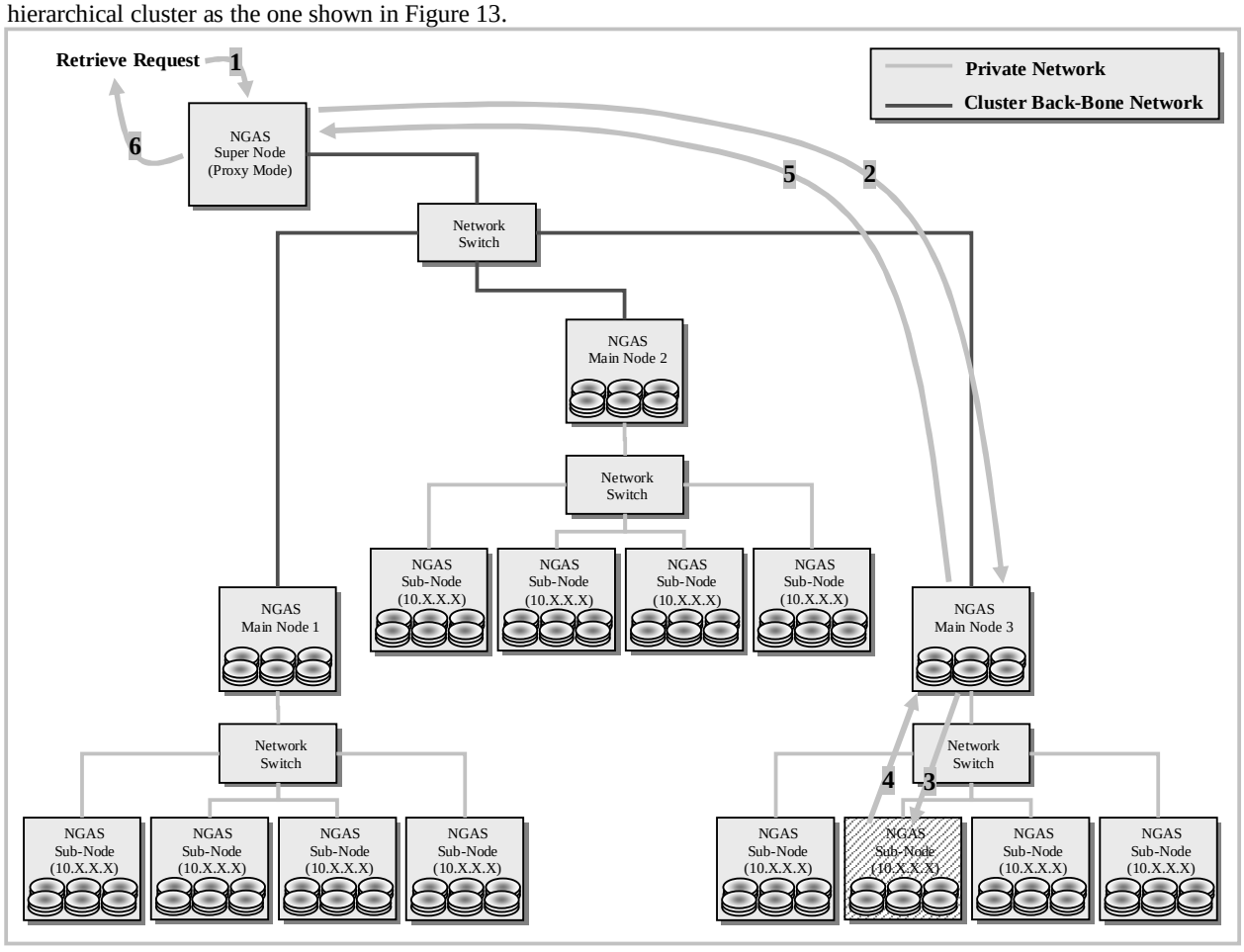

Figure 13: Example of hierarchical NGAS Cluster.

In the cluster shown in Figure 13, the main entry point of the NGAS Cluster (and the only one for that matter), is the NGAS Super Node. All requests must pass through this node. When a Retrieve Request is received by the super node (1), it will identify that the file requested is located on the NGAS Node high-lighted in the figure. This it finds out from the "ngas_files" and "ngas_disks" tables in the NGAS DB (DB: "ngas_files.disk_id" → "ngas_disks.host_id"). From the IP address of the host of interest (DB: "ngas_hosts.ip_address"), the NG/AMS Server on the super node determines that the file cannot be accessed directly since the target node is located within a private network. From the "ngas_hosts" table it finds the NGAS Main Node for that sub-cluster (NGAS Main Node = "ngas_hosts.cluster_name"), and it forwards the Retrieve Request to the NGAS Main Node 3 (2). This in turn figures out that the file is located on a disk hosted in a node within 'its' private network. It therefore retrieves the file via the private network (3 and 4). If processing was requested this

is carried out on the sub-node. Subsequently the NGAS Main Node 3 sends back the final result to the NGAS Super Node (5). The NGAS Super Node in turn, returns the result of the Retrieve Request to the external requestor (6).

The scenario in Figure 13, shows a rather complex environment. The example serves mainly to illustrate the capabilities of the NG/AMS SW. It is probably always an advantage to have one single entry point to an NGAS Archive to make it easy for external clients to access the data. In addition, for security reasons it is an advantage to have only one such entry point to the archive cluster. A disadvantage of this scenario is that each request for data will have to pass through two NG/AMS Server acting as proxies before arriving to the client. This of course means an extra overhead.

If the Suspension/Wake-Up Service is used (see Section 4.3), it is important that each suspended host, is accessible by one other NGAS Host, which is never suspended and therefore can be requested to wake up such a suspended host. In the example in Figure 13, the sub-nodes could be suspended, whereas the main nodes will have to be kept running with an NG/AMS Server in Online State running on them.

A more simple example of an NGAS Cluster is shown in Figure 14.

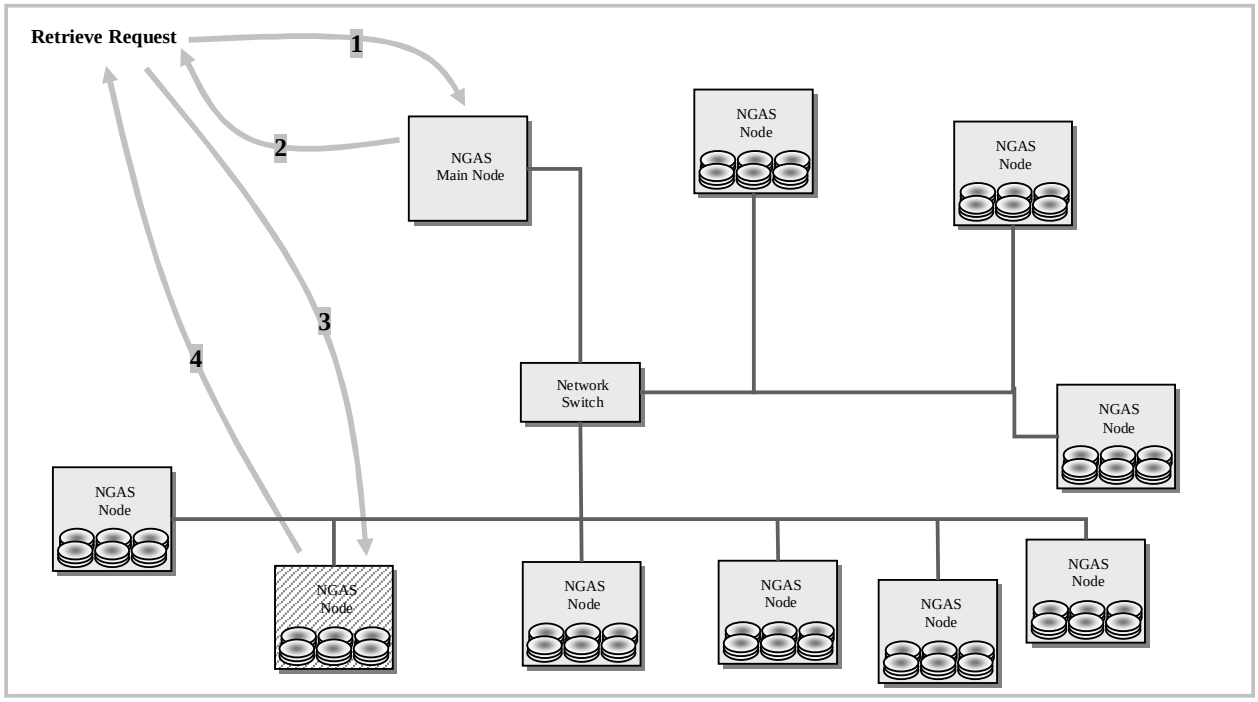

Figure 14: Example of a 'simple' NGAS Cluster.

The architecture in Figure 14 is based on a 'flat structure' providing *external* access to the individual node in the cluster. The client has still one contact point as in the scenario in Figure 13, namely the NGAS Main Node and it sends all Retrieve Requests to this node (1). In this case however, the NGAS Main Node does not act as proxy, and after identifying on which NGAS Node the file is located, it returns an HTTP Redirection Response to the requestor (2). The client now issues the same Retrieve Request directly to the NGAS Node where the file is located (3). The NG/AMS Server on that host handles the request and possibly processes the file and sends this back, *directly* to the requestor (4).

It goes without saying, that the structure shown in Figure 14 makes the handling of requests far more efficient compared to the structure used in Figure 13. There is however still the issue of security to take into account. I.e. all nodes are accessible externally. In addition, the file access is no-longer transparent, since the client has to support the re-direct scheme defined by the HTTP protocol. Using the C or Python-APIs or command utilities however, this is handled transparently for the client (see the Chapters 9 and 10, and Section 5.1).

As a last example scenario an architecture similar to the one in Figure 14 could be used, whereby there is one NGAS Main Node acting as main entry point. All nodes however, in the cluster are connected to the switch via a private network, which is not accessible externally. This topology has the advantage of still having only one contact point for external clients. At the same time access to the data is handled transparently, as the NGAS Main Node will always act as proxy on behalf of

the client. There is only one intermediate copy created (due to the proxy mode) using this architecture (as opposed to two proxy copies in the scenario in Figure 13). The processing is carried out on the individual NGAS Nodes.

Many other architectures can be designed using the NG/AMS Server and NGAS Nodes in various configurations. The three architectures discussed in this section, merely serve to give an impression of what possibilities are provided.

4.2 *EXPERT:* Data Subscription Service

The Data Subscription Service of NG/AMS, makes it possible to synchronize data holdings of different NGAS Nodes partially or completely and to export data on-the-fly from one NGAS archive to remote sites that need part of the data or all data becoming available on an NGAS Host. A client subscribing for data is referred to as a Data Subscriber. An NG/AMS Server, which delivers data to such a Subscriber, is referred to as a Data Provider.

When a client subscribes, it can specify to receive data from a certain point in time. In this way it is possible for a client to receive older files. Otherwise, the time for subscription is taken as starting point and only new files archived from the time of the subscription are taken into account for that client.

It is also possible to specify a Filter Plug-In (see Chapter 21), which is applied on the data files to determine whether or not to deliver the file to a specific Data Subscriber. A client subscribes itself by issuing a SUBSCRIBE command (see also Section 27.13).

The client subscribes itself giving a so-called Subscriber URL to the Data Provider NG/AMS. NG/AMS delivers data to the client by performing an HTTP POST on the Subscriber URL. It is up to the client to specify a proper Subscriber URL. On the client side a corresponding HTTP server must be ready to handle the data delivery POST requests from the Data Provider. Any WEB server can be used, from a simple customized implementation to an existing and widely used server like e.g. Apache[4]. The server must of course be capable of handling the data delivery request. The handling could be implemented as a CGI script.

An NG/AMS Server can be configured to subscribe itself as a Data Subscriber to another NG/AMS Server. In this case the Subscriber URL should be the URL used when performing an Archive Push Request, i.e. "http://<host>:<port>/ARCHIVE" and the corresponding DAPI should be made available within the context of the Data Subscriber NG/AMS Server to handle the possible types of data that can be delivered. In order to make an NG/AMS Server subscribe itself, the configuration must be adjusted accordingly (CFG: "NgamsCfg.SubscriptionDef", see Chapter 6). It is possible to instruct an NG/AMS Server to un-subscribe itself automatically when it goes Offline (CFG: "NgamsCfg.SubscriptionDef:AutoUnsubscribe".

When a client subscribes, it can allocate a priority to itself. This priority determines how much CPU time the delivery of data files to that client may consume. A client that subscribes itself with a lower priority than other Subscribers, will receive the files later than these other Subscribers. It should be evaluated carefully for each client how soon the data should be delivered. The default priority is 10. The lower the priority number, the more CPU time the client is allocated. I.e., in principle "0" would be the highest available priority. It is advisable to allocate such a priority with great care since the data delivery might consume a lot of CPU time, and may interfere with more urgent Archive or Retrieve Requests.

When a client has first subscribed itself to a certain type of data, NG/AMS guarantees that all files of that type and matching the time constraint, will be delivered to the client. If it is impossible to deliver a file if e.g. the client has terminated execution or due to interruption of the network connection, NG/AMS will back-log buffer the data in the Subscription Back-Log and try periodically to deliver the data to the client. Even if the Storage Media hosting the files to deliver to the client are removed from the Data Provider NGAS Host, the files will be delivered, since the Subscription Back-Log is located in a separate area. Note that normally the Back-Log Area should be located on one of the permanent disks of the NGAS Host to facilitate this scheme. *Beware that if files cannot be delivered during a longer period of time, the back-log storage area may fill up with Back-Log Buffered files*. It is possible to specify an expiration period of time indicating for how long time data should be kept in the Subscription Back-Log (CFG: "NgamsCfg.SubscriptionDef.-BackLogExpTime"). Data residing longer than the expiration time, will be deleted and thus never delivered. The name of the Subscription Back-Log is as follows: "<NgamsCfg.Ngams:BackLogBufferDirectory>/subscr-back-log". A table in the NGAS DB is used to keep track of this Subscription Back-Log (see Section 8.5).

A simple scheme has been implemented to avoid that the same data file is delivered several times to a client. This scheme is based on recording the ingestion date for the last file delivered. I.e., only files with a more recent ingestion date will be

[4] http://www.apache.org.

taken into account. This remembered 'last ingestion date' for each client will be reset if a start date for the subscription 'older' than this date is specified by a client.

A Data Subscriber unsubscribes itself by sending an UNSUBSCRIBE command. The client remains subscribed as soon as it has sent the SUBSCRIBE command until an UNSUBSCRIBE is submitted. When a client issues the UNSUBSCRIBE command, the Subscription Back-Log for that client will be reset and thus possible Back-Log Buffered data will not be delivered.

Great care should be taken to avoid 'circular subscriptions', i.e., that two clients subscribe to each other for the same type of data. In such a case, the two serves would continue to deliver the file to each other, ending up saturating the system. A Subscriber cannot subscribe to itself.

It is possible to switch off the Subscription Service globally via the configuration (CFG: "NgamsCfg.SubscriptionDef: Enable). The subscription service is handled by an internal thread (Data Subscription Thread) running within the NG/AMS Server. It is possible to specify how often this thread should be scheduled in the configuration (CFG: "NgamsCfg.SubscriptionDef:SuspensionTime"). This suspension time determines how often the server will try to deliver Subscription Back-Log data. The Data Subscription Thread is scheduled explicitly when new data become available on an NGAS Host. The suspension time, defines how frequently the thread should try to deliver Subscription Back-Log Buffered data.

4.3 *EXPERT:* Server Suspension/Wake-Up Service

Since an NGAS Host may be idling for longer period of times, it is relevant to suspend such a host. This is relevant, in particular in case of clusters of NGAS nodes, which consume a non-negligible amount of power. A feature is provided by NG/AMS whereby it is possible to configure an NG/AMS host to suspend itself after a certain period of idle time (CFG: "NgamsCfg.HostSuspension:IdleSuspensionTime"). Host suspension can be enabled/disabled globally via the configuration (CFG: "NgamsCfg.HostSuspension:IdleSuspension").

When an NG/AMS Server identifies that it should suspend itself, it invokes the so-called Suspension Plug-In (see Chapter 19), which actually takes care of suspending the system. Apart from various/possible clean-up of the system, this usually simply means to shut down the NGAS Host. The host should normally be configured such that when a shut-down is performed, the NG/AMS Server is terminated in a clean manner.

After suspending itself, an NGAS Host can only be 'woken up' by 'external intervention'. This means that either the host must be switched on manually, or the server must request to receive a 'wake-up call' from another NGAS Host. An NGAS Host suspending itself, signals by which other NGAS Host it would like to be woken up (CFG: "NgamsCfg.HostSuspension:WakeUpServerHost"). This means that in an NGAS Cluster where host suspension is used, one host should be kept switched on with an NG/AMS Server running in Online State. An NG/AMS Server suspending itself, will calculate when it should be woken up at latest. This is determined by the time for scheduling the next data checking batch if Data Consistency Checking is active.

A suspended server will also be woken up, if a request for data located on the suspended host is received. In order for this to work, all Retrieve Requests must pass through one node in the NGAS Cluster (main node), which is never suspended. The main node will identify that the requested data is stored on a suspended host, and will wake up this node as described above. Handling a Retrieve Request of data stored on a suspended host, may therefore take some time depending on how long time it takes the host to become operational (Online). A proper time-out must therefore be applied when retrieving data from an NGAS Cluster where host suspension is used. Once the suspended server is Online, requests will be handled rapidly, until it is suspended again (after the specified period of idle time).

An NG/AMS Server, which is requested to wake-up a suspended NGAS Host, will invoke a Wake-Up Plug-In when the time for waking up the suspended host has arrived (CFG: "NgamsCfg.HostSuspension:WakeUpPlugIn"); see also Chapter 20 for more information about the Wake-Up Plug-In. The Wake-Up Plug-In will usually inform some device connected to the network to switch on the suspended NGAS Host. After having launched the Wake-Up Plug-In, the NG/AMS Server will wait for the suspended NGAS Host to become active. If this does not occur within a certain time-out (CFG: "NgamsCfg.HostSuspension:WakeUpCallTimeOut") an error message is logged and an Error Email Notification Message send to the subscribers of this.

5 The NG/AMS Server and Utilities

Three 'executables' are provided within the NG/AMS package. These are 1) The NG/AMS Server - "ngamsServer(.py)", 2) The NG/AMS Python Client - "ngamsPClient(.py)", and 3) The NG/AMS C Client - "ngamsCClient". These executables are described in the following sections.

5.1 NG/AMS Server Command Line Interface

By calling the NG/AMS Server without command line parameters (or illegal ones), the following online help is printed on "stdout":

```
ngasmgr@acngast1:/opsw/NGAS/ngams/ngamsServer>  python ngamsServer.py -h
Correct usage is:

ngamsServer -cfg <cfg file> [-v <level>] [-version] [-license]
            [-locLogFile <log file>] [-locLogLevel <level>
            [-sysLog <level>] [-sysLogPrefix <prefix>]
            [-force] [-autoOnline] [-d]

-cfg <cfg file>        NG/AMS Configuration File.

-v <level>             Verbose Mode + Level.

-version               Print out version of server.

-license               Print out license information.

-locLogFile <file>     Name of Local Log File.
-locLogLevel <level>   Level for logging in Local Log File.

-sysLog                Switch syslog logging on.
-sysLogPrefix <prefix> Prefix for syslog logging.


-d                     Debugging Mode.

-force                 Force execution eventhough PID File found.

-autoOnline            Bring the server to Online State
                       automatically after initialization.

-noAutoExit            If -autoOnline is specified and an
                       error occurs preventing the system
                       from going Online, it will not auto-
                       matically exit.

Note: The values given on the command line, overwrites the
      ones given in the NG/AMS Configuration File.

(c) ESO/DMD 2001-2002 - NGAS Project - http://archive.eso.org/NGAST
ngasmgr@acngast1:/opsw/NGAS/ngams/ngamsServer>
```

Figure 15: The NG/AMS Server online help output (as written on "stdout").

5.2 Python and C Command Utilities

By invoking the NG/AMS Python and C command utilities without input parameters (or with illegal ones), the following online help is written on "stdout" (example help page generated invoking the C Command Utility):

```
> ngamsCClient -host <host> -port <port> -cmd <command> [-v <level>] [-version>] [-d]
    [-noWait] [-status] [-license] [-timeOut <secs>] [<cmd spec pars>]


-host <host>:            Host where the NG/AMS Server is running.

-port <port>:            Port number used by the NG/AMS Server.

-cmd <command>:          Command to issue. The commands accepted by the
                         tool are:

                           ARCHIVE:       Archive a file.
                           CLONE:         Clone a file or a set of file.
                           EXIT:          Make the server exit.
                           INIT:          Re-initialize the server.
                           LABEL:         Print a disk label.
                           ONLINE:        Bring server to Online State.
                           OFFLINE:       Bring server to Offline State.
                           REGISTER       Register a file or a set of files.
                           REMDISK:       Remove information about a disk from
                                          the NGAS DB and delete the files
                                          stored on the disk.
                           REMFILE:       Remove info about a file or a set of
                                          files from the NGAS DB and delete the
                                          files from the disks hosting them.
                           RETRIEVE:      Retrieve a data file from NGAS.
                           STATUS:        Query status.
                           SUBSCRIBE:     Subscribe to data and act as
                                          HTTP daemon that is ready to receive
                                          the data.
                           UNSUBSCRIBE:   Cancel a subscription to data.

-v <level>:              Verbose output level.

-version:                Print version and exit.

-d:                      Run in Debug Mode.

-noWait:                 Don't wait for the NG/AMS Server to terminate the
                         handling of the command.

-status:                 Dump the status message sent by the NG/AMS
                         Server to stdout.

-license:                Print out NG/AMS license information in stdout.

-timeOut <secs>:         Timeout in seconds to apply during the communication
                         with the server. Note: This is only supported for
                         the NG/AMS C-Client. For the P-Client this parameter
                         is ignored.

<cmd spec pars>:         The command specific parameters are:

-diskId <disk ID>:       Refers to a specific disk.

-execute:                Execute an action.

-fileId <id>:            Refers to a specific file.

-fileUri <uri>:          URI pointing to the file to be archived.

                           file:/home/data/Image1.fits

                         This will result in an Archive Pull Request. It can
                         also be given directly as a filename, e.g.:

                           /home/data/Image1.fits
```

```
                        This will result in an Archive Push Request.

-fileVersion <version>:  Refers to a specific File Version of a file.

-filterPlugIn ...:       Reference to a Filter Plug-In to apply in connection
                         with handling the request.

-force:                  Force an action.

-internal <filename>:    Refer to an NG/AMS 'internal file'.

-mimeType <mt>:          If it is not possible to determine the mime-type of
                         a data file from the filename, the mime-type must
                         be explicitly given with the ARCHIVE command.

-ngLog:                  Refers to the Local Log File used by NG/AMS.

-outputFile ...:         Directory or file in which to dump a file when
                         retrieving files.

-path <path>:            Specifies a path, which should be taken into
                         account when carrying out the request.

-plugInPars ...:         Parameters to provide to the plug-in.

-priority <prio>:        Priority. Used by Subscribers to give another
                         priority than the default one (10). A low number
                         means a high priority.

-processing <DDPI>:      Name of Data Processing Plug-In to execute on the
                         data before sending it back to the requestor.

-processingPars <pars>:  Optional parameters to be handed over to the DPPI.
                         These should normally be given on the format:

                           par1=val,par2=val2,...

                         - although it is up to the DPPI to interpret these.

-slotId <id>:            Slot ID for which to generate label.

-startDate ...:          Date in ISO8601 format indicating a lower limit
                         in time.

-url <url>:              URL. Used for the SUBSCRIBE command to indicate
                         to where NG/AMS should deliver data subscribed for.


The parameter combinations valid for each command are:

ARCHIVE:                 -fileUri <uri> [-mimeType <mt>] [-noVersioning]

CLONE:                   -fileId <id> v| -diskId <id> v| -fileVersion <ver>

EXIT:                    None.

INIT:                    None.

LABEL:                   -slotId <id>

OFFLINE:                 [-force]

ONLINE:                  None.

REGISTER:                -path <path>

REMDISK:                 -diskId <id> [-execute]

REMFILE:                 [-diskId <id>] -fileId <id> [-fileVersion <ver>]
                         [-execute]

RETRIEVE:                -fileId <id> [-fileVersion <ver>] [-outputFile <of>]
                         [-processing <DDPI> [-processingPars <pars>]] |
                         -internal <filename> | -ngLog
```

```
STATUS:                 None.

SUBSCRIBE:              -url <url> [-priority <prio>]
                        [-startDate <start date (ISO8601)>]
                        [-filterPlugIn <filter plug-in>
                         [-plugInPars <filter plug-in pars>]]

UNSUBSCRIBE:            -url <url>


(c) ESO/DMD 2001-2002 - NGAS Project - http://archive.eso.org/NGAST
```

Figure 16: The NG/AMS C-Client and Python-Client online help output (on “stdout”).

6 *EXPERT:* Configuring NG/AMS

The NG/AMS SW is implemented in a very flexible way to be able to adjust the system for various scenarios. In order to obtain this, a wide range of parameters can be adjusted in the NG/AMS Configuration.

This chapter contains a description of these parameters. The format for the NG/AMS Configuration also includes the Header Element which is a generic standard header for XML documents. This header is not described here. An example of an NG/AMS Configuration can be found in Section 6.3. The DTDs defining the format of the NG/AMS Configuration can be found in the Sections 6.1 and 6.2.

6.1 *EXPERT:* NG/AMS Configuration DTD - "ngamsCfg.dtd"

The DTD for the NG/AMS Configuration is based on the "ngamsInternal.dtd" (see Section 6.2), which defines the NG/AMS specific elements used in the NG/AMS Configuration. The contents is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % XmlStd SYSTEM "http://www.eso.org/projects/esoxml/XmlStd.dtd">
%XmlStd;
<!ENTITY % NgamsInternal SYSTEM "ngamsInternal.dtd">
%NgamsInternal;

<!--
  E.S.O.

  "@(#) $Id: ngamsCfg.dtd,v 1.7 2002/12/02 18:05:12 arcsw Exp $"

  Who        When        What
  ********   **********  **************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  ngamsCfgNau.dtd defines the contents and lay-out of the
  configuration file loaded by the NG/AMS Server at start-up.

  Consult the DTD ngamsInternal.dtd for further information. It contains the
  actual definition of the elements of the NG/AMS Configuration.
  -->

<!--
  The NgamsCfg element is the root element of the NG/AMS
  Configuration for NG/AMS NAU Systems.
  -->
<!ELEMENT NgamsCfg (Header,
                    (Ngams, Db, MimeTypes, StorageSet*,
                     Stream*, Processing, Register, FileHandling, Monitor,
                   Log, Notification?, HostSuspension))>

<!-- oOo -->
```

Figure 17: NG/AMS Configuration DTD (FILE: "ngams/ngamsData/ngamsCfg.dtd").

6.2 *EXPERT:* NG/AMS Base DTD - "ngamsInternal.dtd"

The base DTD is used to define various XML elements, which can be re-used in various deducted DTD/XML documents. The contents is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  E.S.O.

  Who        When        What
  ********   **********  **************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  The ngamsInternal.dtd defines various common elements to be used
  in the NG/AMS XML documents.
  -->
```

```
<!--
  The Ngams Element defines properties for the individual
  installation of NG/AMS.

  Attributes:
    ArchiveName:            Name of the archive - e.g. ESO-VLT.

    PortNo:                 Port number to use for the NG/AMS HTTP server.

    SwVersion:               If defined, NG/AMS will compare the value of
                            this attribute with the version ID of NG/AMS.
                            If these are not completely identical an error
                            is reported and the NG/AMS Server will perform
                            an exit. If the SwVersion is not specified
                            (set to ""), this check is not done.

    OnlinePlugIn:           Plug-In utility invoked by NG/AMS when
                            going Online to prepare the system and to
                            obtain the information about the current
                            disk configuration and status of the disks.

    OnlinePlugInPars:       Input parameters to the Online Plug-In.

    OfflinePlugIn:          Plug-In utility invoked by NG/AMS when
                            going Offline to prepare the system for
                            standby mode.

    OfflinePlugInPars:      Input parameters to the Online Plug-In.

    Replication:            Indicates if file replication should be
                            carried out by this NG/AMS (0|1).

    BlockSize:              Block size applied when receiving and
                            sending data via HTTP (bytes).

    Simulation:             Simulation system "1" otherwise "0".

    MountRootDirectory:     Base directory used as root directory when
                            mounting the disks.

    AllowArchiveReq:        Allow clients to issue Archive Requests
                            to the system (0|1).

    AllowRetrieveReq:       Allow clients to retrieve data from this
                            NG/AMS installation (0|1).

    AllowProcessingReq:     Allow clients to perform processing requests
                            on this NG/AMS installation (0|1).

    AllowRemoveReq:         Allow clients to carry out REMFILE and REMDISK
                            commands (0|1).

    ForceProxyMode:         If a server which is contacted with a Retrieve
                            Request, finds a file to be retrieved, on
                            another NGAS Host, it will act as proxy and
                            send back the file to the requestor if
                            possible.

    JanitorSuspendTime:     Period of time the Janitor Thread is suspended
                            after each iteration. Should be given as
                            '<days>T<hours>:<minutes>:<seconds>'.

    BackLogBuffering:       Enable/disable Back-Log Data Buffering (0|1).

    BackLogBufferDirectory: Directory that will host the "back-log" directory
                            where data files are buffered for later handling
  -->
<!ELEMENT Ngams EMPTY>
<!ATTLIST Ngams ArchiveName           CDATA         #REQUIRED
                PortNo                CDATA         #REQUIRED
                SwVersion             CDATA         ""
                OnlinePlugIn          CDATA         #REQUIRED
                OnlinePlugInPars      CDATA         #REQUIRED
                OfflinePlugIn         CDATA         #REQUIRED
                OfflinePlugInPars     CDATA         #REQUIRED
                LabelPrinterPlugIn    CDATA         #IMPLIED
                LabelPrinterPlugInPars CDATA        #IMPLIED
                Replication           CDATA         #IMPLIED
                BlockSize             CDATA         #REQUIRED
                Simulation            (0|1)         "0"
                MountRootDirectory    CDATA         #REQUIRED
                AllowArchiveReq       (0|1)         #REQUIRED
                AllowRetrieveReq      (0|1)         #REQUIRED
```

```
               AllowProcessingReq     (0|1)          #REQUIRED
               AllowRemoveReq         (0|1)          #REQUIRED
               ForceProxyMode         (0|1)          #IMPLIED
               JanitorSuspendTime     CDATA          #REQUIRED
               BackLogBuffering       (0|1)          #IMPLIED
               BackLogBufferDirectory CDATA          #IMPLIED>


<!--
  The Db Element defines properties for the interaction
  with the NGAS DB.

  Attributes:
    Server:    Name of DB server.

    Name:      Name of the DB to use.

    User:      The DB user name to connect as.

    Password:  The password for the DB user.
  -->
<!ELEMENT Db EMPTY>
<!ATTLIST Db Server       CDATA  #REQUIRED
             Name         CDATA  #REQUIRED
             User         CDATA  #REQUIRED
             Password     CDATA  #REQUIRED>


<!--
  The MimeTypes Element contains a mapping between the mime-types used
  by NG/AMS and the extension names of the data files.

  The element MimeTypeMap contains the mapping between each mime-type
  and the corresponding extension.

  Attributes:
    MimeType:      Mime-type, e.g. "ngas/fits".

    Extension:     Extension of data file, e.g. "fits" (without
                   leading dot).
  -->
<!ELEMENT MimeTypes (MimeTypeMap+)>
<!ELEMENT MimeTypeMap EMPTY>
<!ATTLIST MimeTypeMap   MimeType   CDATA  #REQUIRED
                        Extension  CDATA  #REQUIRED>


<!--
  The StorageSet Element defines properties for each 'storage unit',
  consisting of two disks (Main Disk and Replication Disk).

  Attributes:
    StorageSetId:       ID used to refer to this particular
                        storage unit (string).

    DiskLabel:          Name of label prefix to put on the
                        disk. Optional (string).

    MainDiskSlotId:     Slot ID for Main Disk (string).

    RepDiskSlotId:      Slot ID for Replication Disk (string).

    Mutex:              Indicates if mutual exclusion access should be
                        enforced to the disks. If this is enabled, only
                        one data reception session will write data to
                        that slot (disk), and not simultaneous as otherwise
                        done if several files for the same disk is received
                        at the same time. 1 = mutual exclusion (integer).

    Synchronize:        The synchronize flag is used to indicate if the two
                        disks of a set should be considerd as locked
                        together so that when one is completed also
                        the other is marked as completed although it
                        may not be (integer/0|1).
  -->
<!ELEMENT StorageSet EMPTY>
<!ATTLIST StorageSet StorageSetId        CDATA  #REQUIRED
                     DiskLabel           CDATA  ""
                     MainDiskSlotId      CDATA  #REQUIRED
                     RepDiskSlotId       CDATA  #IMPLIED
                     Mutex               (0|1)  "0"
                     Synchronize         (0|1)  "1">


<!--
```

```
  The FileHandling Element defines properties for the file handling.

  Attributes:
    ProcessingDirectory:  Base directory in which temporary files
                          will be stored during processing.

    PathPrefix:           A prefix which will be used when building up the
                          target filename. Could e.g. be used as to build
                          a filename like:

                          <mount point>/<PathPrefix>/<add. path>/<filename>

                          - but it is up to the Data Handling Plug-In
                          to use it in a way appropriate for the context.

    ChecksumPlugIn:       DCPI - Data Checksum Plug-In, which generates the
                          checksum of a data file.

    ChecksumPlugInPars:   Input parameters to the DCPI.

    DataCheckActive:      Enable/disable Data Check Service (1 = activate).

    DataCheckPrio:        Priority of Data Check Thread (1 = highest priority).
                          with this parameter it is possible to adjust the
                          amount of CPU power the Data Check Thread should
                          consume. The higher the number, the lesser CPU
                          the check will consume. At the same time, it
                          will take more time to carry out the check of
                          the data holding.

    DataCheckMinCycle:    Indicates the minimum allowed time for carrying out
                          one check cycle (ddThh:mm:ss). If this is specified
                          e.g. to 24 hours, and one check cycle would only
                          take 11 hours, the check would be suspended for
                          13 hours before running again.

    DataCheckDiskSeq:     Used to indicate if disks should be checked
                          sequentially (ordered according to the Slot
                          ID), or randomly (SEQUENTIAL|RANDOM).

    DataCheckFileSeq:     Used to indicate if files on a disk should be checked
                          sequentially (ordered according to the ingestion
                          date), or randomly (SEQUENTIAL|RANDOM).

    DataCheckLogSummary:  If set to 1, a summary log info will be generated
                          each time a complete check of the data holding
                          of one NGAS Host has been carried out.
  -->
<!ELEMENT FileHandling EMPTY>
<!ATTLIST FileHandling ProcessingDirectory  CDATA               #REQUIRED
                       PathPrefix           CDATA               #REQUIRED
                       ChecksumPlugIn       CDATA               #REQUIRED
                       ChecksumPlugInPars   CDATA               #REQUIRED
                       DataCheckActive      (0|1)               #REQUIRED
                       DataCheckPrio        CDATA               #REQUIRED
                       DataCheckMinCycle    CDATA               #REQUIRED
                       DataCheckDiskSeq     (SEQUENTIAL|RANDOM) #REQUIRED
                       DataCheckFileSeq     (SEQUENTIAL|RANDOM) #REQUIRED
                       DataCheckLogSummary  (0|1)               #REQUIRED>

<!--
  The Stream Element defines properties for the handling of data
  streams into the NGAS System.

  The StorageSetRef contains an attribute with a reference to a StorageSet.

  Attributes:
  MimeType:      Mime-type identifying this type of data.

  PlugIn:        A plug-in command that will be executed to 1) Check the
                 consistency of the data file of the given mime-type,
                 2) Generate the necessary information for the NGAS DB.

  PlugInPars:    Parameters which will be transferred to the plug-in
                 function.
  -->
<!ELEMENT Stream (StorageSetRef+)>
<!ATTLIST Stream MimeType       CDATA  #REQUIRED
                 PlugIn         CDATA  #REQUIRED
                 PlugInPars     CDATA  #IMPLIED>
<!ELEMENT StorageSetRef EMPTY>
<!ATTLIST StorageSetRef  StorageSetId CDATA  #REQUIRED>


<!--
```

```
  The Processing Element defines the set of DPPIs that are available
  on an NGAS system.

  The element contains a number of PlugIn Elements which each define
  the properties of the DPPIs. In addition the PlugIn Element defines
  which types of data can be handled.
-->
<!ELEMENT Processing (PlugIn*)>
<!ELEMENT PlugIn (MimeType+)>
<!ATTLIST PlugIn Name         CDATA  #REQUIRED
                 PlugInPars   CDATA  #REQUIRED>
<!ELEMENT MimeType  EMPTY>
<!ATTLIST MimeType  Name      CDATA  #REQUIRED>


<!--
  The Register Element defines the of DRPIs needed for registering
  various file types.
-->
<!ELEMENT Register (PlugIn*)>


<!--
  The Monitor Element defines properties for ingestion/handling
  of data files.

  Attributes:
    MinFreeSpaceWarning:    Indicates the free disk space limit before issuing
                            an Warning Log Message (MB).

    FreeSpaceDiskChangeMb:  Indicates the free disk space limit before
                            changing disk (MB).
  -->
<!ELEMENT Monitor EMPTY>
<!ATTLIST Monitor MinFreeSpaceWarningMb  CDATA #REQUIRED
                  FreeSpaceDiskChangeMb  CDATA #REQUIRED>


<!--
  The Log Element defines properties for the logging performed by
  the NG/AMS Server.

  Attributes:
    SysLog:         Switch on UNIX syslog logging (0|1).

    SysLogPrefix:   Prefix (tag) written first in the syslog entries
                    (in the data part).

    LocLogFile:     Indicates a name of a local log file. Should be
                    complete path.

    LocalLogLevel:  Log level for producing logs into the local
                    log file. Must be in the interval from 0 to 5.

    LogBufferSize:  Size of the internal log buffer (in terms of log
                    entries). This indicates how many entries are buffered
                    before the logs are purged out to the log file.
  -->
<!ELEMENT Log EMPTY>
<!ATTLIST Log SysLog          (0|1)         #REQUIRED
              SysLogPrefix    CDATA         #REQUIRED
              LocalLogFile    CDATA         #REQUIRED
              LocalLogLevel   (0|1|2|3|4|5) #REQUIRED
              LogBufferSize   CDATA         #REQUIRED>


<!--
  The Notification Element is used to define subscribers (e-mail
  recipients) that will receive an e-mail when certain events
  occur. For the moment the following events can trigger e-mails
  to one or more recipients:

    o Alert Logs.
    o Error Logs.
    o Disk Running Full (Min. Free Disk Space).
    o Disk Change (Disk Change Log).
    o No more disk space.
  -->
<!ELEMENT Notification (AlertNotification?, ErrorNotification?,
                        DiskSpaceNotification?, DiskChangeNotification?,
                        NoDiskSpaceNotification?,
                        DataErrorNotification?)>
<!ATTLIST Notification  SmtpHost   CDATA  #REQUIRED
                        Active     (0|1)  #REQUIRED>
<!ELEMENT EmailRecipient EMPTY>
```

```
<!ATTLIST EmailRecipient Address   CDATA  #REQUIRED>
<!ELEMENT AlertNotification (EmailRecipient+)>
<!ELEMENT ErrorNotification (EmailRecipient+)>
<!ELEMENT DiskSpaceNotification (EmailRecipient+)>
<!ELEMENT DiskChangeNotification (EmailRecipient+)>
<!ELEMENT NoDiskSpaceNotification (EmailRecipient+)>
<!ELEMENT DataErrorNotification (EmailRecipient+)>


<!--
  The Host Suspension Element is used to define the conditions for
  suspending an NGAS Host and for waking up a suspended NGAS Host.

  Attributes:
      IdleSuspension:      Switches on/off host idle suspension (0 = Off,
                           1 = 0n).

      IdleSuspensionTime:  The idle time that must elapse for an  NGAS
                           system to suspending itself given in seconds.

      WakeUpServerHost:    Name of server that should be requested to
                           wake up a suspended NGAS Host.

      SuspensionPlugIn:    Name of plug-in to be invoked to suspend the
                           NGAS Host.

      WakeUpPlugIn:        Name of plug-in to be invoked to wake-up a
                           suspended NGAS Host.

      WakeUpCallTimeOut:   Time-out for waiting for an NGAS Host being
                           woken to wake up.
  -->
<!ELEMENT HostSuspension EMPTY>
<!ATTLIST HostSuspension  IdleSuspension       (0|1)  "0"
                          IdleSuspensionTime   CDATA  "0"
                          WakeUpServerHost     CDATA  ""
                          SuspensionPlugIn     CDATA  ""
                          WakeUpPlugIn         CDATA  ""
                          WakeUpCallTimeOut    CDATA  "0">


<!--
  The SubscriptionDef Element is used to define the properties for
  Data Subscription. It may contain a number of Subscription Elements
  indicating to which server this NG/AMS should subscribe to.

  Attributes:
      AutoUnsubscribe:    Indicates if the server should un-subscribe
                          automatically when going Offline.

      SuspensionTime:     Time the thread should suspend itself between
                          attempting to deliver back-logged buffered
                          files (ISO8601).

      BackLogExpTime:     Expiration time for entries in the
                          Subscription Back-Log (string/ISO8601).
  -->
<!ELEMENT SubscriptionDef (Subscription*)>
<!ATTLIST SubscriptionDef AutoUnsubscribe   (0|1)  "1"
                          SuspensionTime    CDATA  #REQUIRED
                          BackLogExpTime    CDATA  #REQUIRED
                          Enable            (0|1)  "1">


<!--
  The Subscription Element define a data subscription. It will trigger
  the NG/AMS Server to subscribe itself to data from another NG/AMS Server.

  Attributes:
      HostId:             Host ID of the Data Provider.

      PortNo:             Port number used by the Data Provider.

      Priority:           Priority requested for this Subscription.
                          The lower the number the higher the priority.
                          Default is 10. If other Subscribers subscribe
                          with a higher priority then this client, they
                          will be served faster.

      SubscriberUrl:      Subscriber URL, which is issued when subscribing.
                          The Data Provider will use this URL when delivering
                          the data to the Subscriber.

      FilterPlugIn:       A Filter Plug-In applied on the data files to
                          decide whether to deliver a file to a Subscriber
```

```
                        or not.

      FilterPlugInPars:   Parameters to issue to the Filter Plug-In.
  -->
<!ELEMENT Subscription    EMPTY>
<!ATTLIST Subscription    HostId              CDATA  #REQUIRED
                          PortNo              CDATA  #REQUIRED
                          Priority            CDATA  "10"
                          SubscriberUrl       CDATA  #REQUIRED
                          FilterPlugIn        CDATA  #IMPLIED
                          FilterPlugInPars    CDATA  #IMPLIED>

<!-- oOo -->
```

Figure 18: NG/AMS Configuration generic DTD (FILE: “ngams/ngamsData/ngamsInternal.dtd”).

6.3 *EXPERT:* NG/AMS Configuration - Example

In the following, an example NG/AMS Configuration is listed:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE NgamsCfg SYSTEM "ngamsCfg.dtd">

<!--
  E.S.O.

  Who        When        What
  ********   **********  ************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  This is the NG/AMS Configuration for the WFI NGAS NCU System.

  Consult the DTDs ngamsCfgNbuNcu.dtd and ngamsInternal.dtd for further
  information.
  -->

<NgamsCfg>

  <Header Name="NgamsCfg.ESOECF.GarNcu1.xml"
          Type="NGAMS-CONFIGURATION-NBU"
          Context="NGAMS"
          Release="1.0"
          Source="jknudstr@eso.org"
          Revision="@(#) $Id: ngamsServer.xml,v 1.1 2003/01/03 10:36:45 arcsw Exp $">
    <Description>
      This XML document contains the configuration for the jewel64 machine
      running as a buffering unit (NCU).
    </Description>
  </Header>

  <Ngams  ArchiveName="ESO-ARCHIVE"
          PortNo="7777"
          SwVersion="v2.0-Beta2"

          OnlinePlugIn="ngamsLinuxOnlinePlugIn"
          OnlinePlugInPars="uri=http://localhost:1080/technical.html,
                            module=3w-xxxx"

          OfflinePlugIn="ngamsLinuxOfflinePlugIn"
          OfflinePlugInPars="unmount=0"

          LabelPrinterPlugIn="ngamsBrotherPT9200DxPlugIn"
          LabelPrinterPlugInPars="dev=/dev/ttyS0,font_file=/opsw/NGAS/ngams/ngamsData/ngamsBrotherPT9200DxFonts.fnt"

          Replication="1"

          BlockSize="65536"

          Simulation="0"
          MountRootDirectory="/NGAS"

          AllowArchiveReq="1"
          AllowRetrieveReq="1"
          AllowProcessingReq="1"
          AllowRemoveReq="1"

          ForceProxyMode="1"

          JanitorSuspendTime="0T00:03:00"

           BackLogBuffering="1"
           BackLogBufferDirectory="/NGAS/ngams_staging"/>

  <Db     Server="LUXSRV"
          Name="ngas"
          User="ngas"
          Password="********"/>

  <MimeTypes>
    <MimeTypeMap  MimeType="image/x-fits"         Extension="fits"/>
    <MimeTypeMap  MimeType="ngas/nglog"           Extension="nglog"/>
    <MimeTypeMap  MimeType="ngas/paf"             Extension="paf"/>
    <MimeTypeMap  MimeType="ngas/log"             Extension="log"/>
    <MimeTypeMap  MimeType="application/x-gfits"  Extension="fits.gz"/>
    <MimeTypeMap  MimeType="application/x-cfits"  Extension="fits.Z"/>
    <MimeTypeMap  MimeType="application/x-hfits"  Extension="hfits"/>
    <MimeTypeMap  MimeType="image/x-fitshdr"      Extension="hdr"/>
  </MimeTypes>
```

```
<StorageSet StorageSetId="FitsStorage1"
            DiskLabel="FITS"
            MainDiskSlotId="1"
            RepDiskSlotId="2"
            Mutex="0"
            Synchronize="1"/>

<StorageSet StorageSetId="FitsStorage2"
            MainDiskSlotId="3"
            RepDiskSlotId="4"
            Mutex="0"
            Synchronize="1"/>

<StorageSet StorageSetId="FitsStorage3"
            DiskLabel="FITS-PAF"
            MainDiskSlotId="5"
            RepDiskSlotId="6"
            Mutex="0"
            Synchronize="1"/>

<StorageSet StorageSetId="FitsStorage4"
            MainDiskSlotId="7"
            Mutex="0"
            Synchronize="1"/>

<Stream  MimeType="image/x-fits"
         PlugIn="ngamsFitsPlugIn"
         PlugInPars="compression=compress -f,checksum_util=utilFitsChecksum,
                     checksum_result=0/0000000000000000">
  <StorageSetRef StorageSetId="FitsStorage1"/>
  <StorageSetRef StorageSetId="FitsStorage2"/>
  <StorageSetRef StorageSetId="FitsStorage3"/>
  <StorageSetRef StorageSetId="FitsStorage4"/>
</Stream>

<Stream  MimeType="application/x-cfits"
         PlugIn="ngamsFitsPlugIn"
         PlugInPars="compression=compress -f,checksum_util=utilFitsChecksum,
                     checksum_result=0/0000000000000000">
  <StorageSetRef StorageSetId="FitsStorage1"/>
  <StorageSetRef StorageSetId="FitsStorage2"/>
  <StorageSetRef StorageSetId="FitsStorage3"/>
  <StorageSetRef StorageSetId="FitsStorage4"/>
</Stream>

<Stream  MimeType="ngas/nglog"
         PlugIn="ngamsNgLogPlugIn"
         PlugInPars="">
  <StorageSetRef StorageSetId="FitsStorage1"/>
  <StorageSetRef StorageSetId="FitsStorage2"/>
  <StorageSetRef StorageSetId="FitsStorage3"/>
  <StorageSetRef StorageSetId="FitsStorage4"/>
</Stream>

<Processing>
  <PlugIn Name="ngamsEsoArchDppi"
          PlugInPars="">
    <MimeType Name="image/x-fits"/>
    <MimeType Name="application/x-gfits"/>
    <MimeType Name="application/x-cfits"/>
  </PlugIn>
  <PlugIn Name="ngamsExtractFitsHdrDppi"
          PlugInPars="">
    <MimeType Name="image/x-fits"/>
    <MimeType Name="application/x-gfits"/>
    <MimeType Name="application/x-cfits"/>
  </PlugIn>
  <PlugIn Name="ngasWfiPreview"
          PlugInPars="">
    <MimeType Name="image/x-fits"/>
    <MimeType Name="application/x-gfits"/>
    <MimeType Name="application/x-cfits"/>
  </PlugIn>
</Processing>

<Register>
  <PlugIn Name="ngamsFitsRegPlugIn"
          PlugInPars="">
    <MimeType Name="image/x-fits"/>
    <MimeType Name="application/x-gfits"/>
    <MimeType Name="application/x-cfits"/>
  </PlugIn>
</Register>

<FileHandling  ProcessingDirectory="/NGAS"
```

```
                PathPrefix="saf"
                ChecksumPlugIn="ngamsGenCrc32"
                ChecksumPlugInPars=""
                DataCheckActive="0"
                DataCheckPrio="1"
                 DataCheckMinCycle="00T00:02:00"
                 DataCheckDiskSeq="SEQUENTIAL"
                 DataCheckFileSeq="SEQUENTIAL"
                DataCheckLogSummary="1"/>

  <Monitor MinFreeSpaceWarningMb="5000"
           FreeSpaceDiskChangeMb="300"/>

  <Log     SysLog="1"
           SysLogPrefix="DFSLog"
           LocalLogFile="/NGAS/ngams_staging/log/LogFile.nglog"
           LocalLogLevel="1"
           LogBufferSize="100"/>

  <Notification SmtpHost="smtphost.hq.eso.org"
                Sender="ngast@eso.org"
                 Active="0">
    <AlertNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </AlertNotification>

    <ErrorNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </ErrorNotification>

    <DiskSpaceNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </DiskSpaceNotification>

    <DiskChangeNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </DiskChangeNotification>

    <NoDiskSpaceNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </NoDiskSpaceNotification>

    <DataErrorNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </DataErrorNotification>
  </Notification>

  <HostSuspension  IdleSuspension="0"
                   IdleSuspensionTime="10"
                   WakeUpServerHost="dmdarc1"
                   SuspensionPlugIn="ngamsTestSuspensionPlugIn"
                   WakeUpPlugIn="ngamsTestWakeUpPlugIn"
                   WakeUpCallTimeOut="10"/>

  <SubscriptionDef AutoUnsubscribe="1"
                    SuspensionTime="0T00:03:00"
                   BackLogExpTime="28T00:00:00"
                   Enable="0">
  </SubscriptionDef>

</NgamsCfg>

<!-- oOo -->
```

Figure 19: Example NG/AMS Configuration file (FILE: “ngams/ngamsData/ngamsServer.xml”).

7 *EXPERT:* NG/AMS Server Communication Protocol

The NG/AMS command interface is based on the HTTP protocol, which is a widely used standard protocol. This makes it easy to interface various client applications with NG/AMS. In this chapter the details of the NG/AMS command interface are described.

Using the NG/AMS Python- and C-APIs, the client applications do not need to worry about the format of the requests sent and replies generated by NG/AMS. It is therefore recommended whenever possible to use the APIs provided with the NG/AMS package.

7.1 *EXPERT:* Format of NG/AMS HTTP Command Messages

The format of the NG/AMS messages is defined as follows:

Archive Push Request:

```
POST ARCHIVE HTTP/1.0[1]
User-Agent: <user agent>
Content-Type: ngas/archive-request | <mime-type>
Content-Length: <length>
Content-Disposition: attachment; filename=<file uri>[; wait=0|1][; no_versioning=0|1]

<data>
```

Figure 20: Format of an Archive Push HTTP request.

Example:

```
POST ARCHIVE HTTP/1.0
User-Agent: NG/AMS C-API
Content-Type: ngas/archive-request
Content-Length: 69120
Content-Disposition: attachment; filename="/tmp/TestFile.fits";wait="1"

~□□‡v}yzy~ƒ}{uˆ~,‰…tcv,, …
```

Figure 21: Example of Archive Push HTTP request.

Archive Pull Request + Other Commands:

```
GET <command>?[<parameter>=<value>] HTTP/1.0
```

Figure 22: Structure of NG/AMS GET method HTTP request.

Example, Archive Pull Request:

```
GET ARCHIVE?filename="file:///tmp/SmallFile.fits"&wait="1" HTTP/1.0
```

Figure 23: Example of NG/AMS GET method HTTP request (Archive Pull Request).

The exact list of parameters for each command are described in Chapter 27.

7.2 *EXPERT:* Format of the NG/AMS HTTP Reply

The format of replies from NG/AMS is defined as follows:

```
HTTP/<HTTP version> <HTTP response code> <message>
Server: <server ID>
Date: <date for generating reply>
Expires: <expiration date (= Date:)>
Content-Type: <mime-type>
Content-Length: <data length>

<data>
```

Figure 24: Format of NG/AMS HTTP response

An example of a reply to an Archive Request is:

```
HTTP/1.0 200 OK
Server: NGAMS/v2.0-Beta2/2002-12-04T09:22:53
Content-type: text/xml
Expires: Mon, 23 Dec 2002 16:10:43 GMT
Content-length: 1188
Date: Mon, 23 Dec 2002 16:10:43 GMT

<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-12-23T16:10:43.079" HostId="acngast1" Message="Successfully handled Archive Push
          Request for data file with URI: SmallFile.fits" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
  <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32300" BytesStored="8567866905" Checksum=""
              Completed="0" CompletionDate="" DiskId="IC35L040AVER07-0-SXPTX093675" HostId="acngast1"
              InstallationDate="2002-11-25T09:48:25.000" LastCheck="" LogicalName="FITS-M-000001"
              Manufacturer="IBM" MountPoint="/NGAS/data1" Mounted="1" NumberOfFiles="164" SlotId="1"
              TotalDiskWriteTime="896.20280099" Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1246906309" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="TEST.2001-05-08T15:25:00.123"
                FileName="saf/2001-05-08/3/TEST.2001-05-08T15:25:00.123.fits.Z" FileSize="53546"
                FileStatus="00000000" FileVersion="3" Format="application/x-cfits" Ignore="0"
                IngestionDate="2002-12-23T16:10:42.000" Tag="" UncompressedFileSize="69120"/>
  </DiskStatus>
</NgamsStatus>
```

Figure 25: Example of NG/AMS HTTP response (Archive Request).

In a reply to a Retrieve Request the data returned will be contained in the message rather than the NG/AMS Status XML document shown above. Such a reply thus looks like this, e.g.:

```
HTTP/1.0 200 OK
Server: NGAMS/v2.0-Beta2/2002-12-04T09:22:53
Content-type: application/x-cfits
Expires: Mon, 23 Dec 2002 16:15:22 GMT
Date: Mon, 23 Dec 2002 16:15:22 GMT
Content-disposition: attachment; filename="TEST.2001-05-08T15:25:00.123.fits.Z"
Content-length: 53546

<data>
```

Figure 26: Example of NG/AMS HTTP response, Retrieve Request.

It is foreseen at a later stage to make it possible to query several files simultaneously with one query. This means that the mime-type "multipart/mixed" will be used as the overall mime-type of the reply and that each part has its proper mime-type defined.

7.3 *EXPERT:* Format of the NG/AMS Redirection HTTP Response

If an NG/AMS Server is not configured to always act as a proxy when data is being requested by a client, HTTP redirection response messages may be generated and send back to the requestor. The format of such redirection responses is:

```
HTTP/1.0 303 Method
Server: <server ID>
Date: <date>
Expires: <date>
Location: <URL pointing to actual location of file>
Content-Type: text/xml
Content-Length: <length>

<NG/AMS status document>
```

Figure 27: Structure of NG/AMS HTTP Redirection Response.

An example of such a redirection HTTP response is:

```
HTTP/1.0 303 Method
Server: v1.5/2002-02-12T10:52:10
Date: Tue,  Jan 01:34:40 2 GMT
Expires: Tue,  Jan 01:34:40 2 GMT
Location: http://jewel64:7777/RETRIEVE?file_id="WFI.2001-09-25T21:19:17.508"
Content-Type: text/xml
Content-Length: 339

<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2001-01-02T01:34:40.656" HostId="jewel68"
          Message="NGAMS_INFO_RETRIEVE_REDIRECT:4024:INFO:
          Redirection URL: http://jewel64:7777/RETRIEVE?file_id=WFI.2001-09-25T21:19:17.508"
          State="ONLINE" Status="SUCCESS" SubState="BUSY" Version="v1.5/2002-02-12T10:52:10"/>
</NgamsStatus>
```

Figure 28: Example of NG/AMS HTTP Redirection Response.

The client must then re-issue the Retrieve Request to the alternative location given in the redirection response and will be able to get access to the data directly from that location (if the system permits). It should be mentioned that it is normally more efficient to request the data directly from the location where it is actually located rather than using NG/AMS as a proxy server. Using the NG/AMS APIs this is all handled transparently for the client application.

8 *EXPERT:* The NGAS DB

The NG/AMS SW is based on four tables in the NGAS DB. These are:

- **ngas_disks**: Contains information about the disks, which have been registered in an NGAS installation.
- **ngas_disks_hist:** Contains a log about major events that has occurred in the life-time of a disk. The information in the "ngas_disks_hist" is newer removed automatically by NG/AMS and will continue to grow with time, whereby the amount of events recorded are kept to a minimum, i.e., only the most essential events in the life-time of a disk are logged in this table. The entries of this table can be used to keep analyze how a disk have been used in the NGAS system, e.g. how many times it has been registered (after 're-cycling'), and when it was registered the first time.
- **ngas_files**: Contains information about each files, which have been archived into NGAS.
- **ngas_hosts**: Contains information about the hosts in an NGAS installation.

In addition two 'internal tables' are used by NG/AMS:

- **ngas_subscr_back_log:** Used by NG/AMS to keep track of files that should have been delivered to a Subscriber whereby the delivery failed.
- **ngas_subscribers:** Contains a persistent snap-shot of the Subscribers that are subscribed to a given NG/AMS Server.

The present release of NG/AMS is based on Sybase ASE, but it is foreseen to be able to make the SW work with other DBMS' by using ODBC.

The system is capable of operating without the availability of the "ngas_hosts" table. In addition it is possible to operate with a DB where the "ngas_hosts" is available but is empty. This however only, under certain conditions. It is however recommended to create and populate this table at least when operating an NGAS Cluster.

Usually it is not foreseen that external applications perform queries directly into the NGAS DB. I.e., all information needed, should be retrieved via the NG/AMS Server. Apart from saving external applications from knowing technical details about the NGAS DB, this has the advantage of making such external applications independent of the DBMS used by an NG/AMS installation. For this reason, it is not guaranteed that 100% backwards compatibility is maintained when it comes to the format of the NGAS DB.

In the following sections the exact contents of the NGAS tables is described.

8.1 *EXPERT:* Table - "ngas_disks"

Column	*Type*	*Value*	*Description*
disk_id	varchar(128)	not null	The ID of the disk. This information is extracted by the Online Plug-In from the BIOS of the disk drive. This is the unique identifier of the disk.
archive	varchar(64)	not null	The name of the archive to which the disk belongs. The value for this is taken from the NG/AMS Configuration.
installation_date	datetime	not null	The date for registering the disk the first time. Subsequent re-registering do not change the value of this column.
type	varchar(64)	not null	Describe the type of the media, e.g.: "MAGNETIC DISK/ATA". The value for this is generated by the Online Plug-In
manufacturer	varchar(64)	null	The manufacturer of the disk. Could e.g. be "IBM" or "Seagate". This value is generated by the Online Plug-In.
logical_name	varchar(128)	not null	The Logical Name of the disk, is a 'human readable' (unique) ID for the disk. It is generated by NG/AMS when the disk is registered the first time.
host_id	varchar(32)	null	The ID of the host where a disk is currently registered. If the disk is not registered in any NGAS Host, this will be set to "".
slot_id	varchar(32)	null	The ID of the slot in the NGAS Host, in which the disk is currently registered. If the disk is not registered, this will be "".
mounted	tinyint	null	Used to indicate if a disk is mounted or not (1 = mounted, 0 - not mounted).

mount_point	varchar(128)	null	Used to give the (complete) name of the mount point where the disk is mounted. If the disk is not mounted this will be "".
number_of_files	int	not null	Indicates how many data files that have been archived on the disk.
available_mb	int	not null	Used to indicate the amount of available storage capacity still free on the disk (given in MBs).
bytes_stored	numeric(20, 0)	not null	Used to indicate the amount of data stored on the disk (given in bytes).
completed	tinyint	not null	Used to indicate that the disk is 'completed', i.e., NG/AMS has been archiving files on the disk, and has reached the threshold specified in the configuration file.
completion_date	datetime	null	Set by NG/AMS when the disk reached the threshold for completion.
checksum	varchar(64)	null	The global checksum value for the disk. ***Note, this is not set for the moment!***
total_disk_write_time	float	null	Total time spent on writing data on the disk.
last_check	datetime	null	Timestamp for when the last check was carried out. This is used to schedule the checking of the data holdings on the disks so that the disks that were not checked or the ones that was checked the longest time ago, are checked first.
last_host_id	varchar(32)	null	The ID of the host were the disk was registered the last time. This is used in order to identify where a file/disk is located even though the host has been suspended.

Table 12: Contents of the NGAS Disks DB Table.

8.2 *EXPERT:* Table - "ngas_disks_hist"

Column	*Type*	*Value*	*Description*
disk_id	varchar(128)	not null	See “disk_id” in “ngas_disks” table.
hist_date	datetime	not null	Timestamp indicating when the event happened in the life-time of the disk.
hist_origin	varchar(64)	not null	The originator of the event, i.e., identification of the NG/AMS hosting the disk.
hist_synopsis	varchar(255)	not null	Short headline indicating the type of event that occurred.
hist_descr_mime_type	varchar(64)	null	Mime-type of the data stored in the history description column.
hist_descr	text	null	Additional information in connection with the event. This will typically be a snap-shot of the NgasDiskInfo file at the time the event occurred. This is e.g. the case when new disks are registered and when disks are removed from the system.

Table 13: Contents of the NGAS Disks History DB Table.

8.3 *EXPERT:* Table - "ngas_files"

Column	*Type*	*Value*	*Description*
disk_id	varchar(128)	not null	ID of the disk where the file is stored.
file_name	varchar(255)	not null	Name of the file. This must be given relative to the mount point of the disk.
file_id	varchar(64)	not null	File ID allocated to the file by the DAPI. The set of File ID, Disk ID and File Version, uniquely defines a file.
file_version	int	default 1	Version of the file. The first version is number 1.
format	varchar(32)	not null	Format of the file. This is generated by the DAPI. Should be the mime-type of the file, as stored on the disk.
file_size	numeric(20, 0)	not null	Size of the file. This must be given in bytes. If the file is compressed, the compressed file size must be given as value for this column.

uncompressed_file_size	numeric(20, 0)	not null	If the file was compressed this indicates the size of the uncompressed file. If the file is not compressed this will be equal to the file_size.
compression	varchar(32)	null	The compression method applied on the file. Could be e.g. "gzip". This should indicate clearly how the file has been compressed, to make it possible to decompress it at a later stage.
ingestion_date	datetime	not null	Date the file was ingested/archived.
ignore	tinyint	null	Used to indicate that this file should be ignored (1 = ignore). If set to one, this entry for this file, will not be taken into account by NG/AMS when files or information about files is queried.
checksum	varchar(64)	null	Checksum of the file. This value is generated by the checksum plug-in specified in the configuration.
checksum_plugin	varchar(64)	null	Name of the checksum plug-in used to generate the checksum for the file. This is used by NG/AMS when performing the Data Consistency Checking of data files. NG/AMS in this way, invokes the same plug-in as was used to generate the checksum originally.

file_status	char(8)	default '00000000'	Current status of the file. The status should be seen as a sequence of bytes, each with a certain signification what concerns the condition and status of the file. These bytes are used to indicate the following (when set to 1). The bytes in the status are counted from left to right:
			
			`1: The File Checksum is incorrect.`
			`2: File being checked.`
			`3-8: Not used.`
			
			The bytes 3-8 may be used at a later stage.

Table 14: Contents of the NGAS Files DB Table.

8.4 *EXPERT:* Table - "ngas_hosts"[5]

Column	*Type*	*Value*	*Description*
host_id	varchar(32)	not null	ID of the NGAS Host, e.g. "jewel65". Should be given only as the name, i.e., without the domain name.
domain	varchar(30)	not null	Domain name of the NGAS Host, e.g. "hq.eso.org".
ip_address	varchar(20)	not null	The IP address of the NGAS Host.
ngas_type	varchar(10)	not null	The type of NGAS Host, i.e., which role it has. The value of this is not used by the NG/AMS SW. Suggested values could be "NAU" - NGAS Archiving Unit, "NBU" - NGAS Buffering Unit, "NCU" - NGAS Central Unit, "NMU" - NGAS Master Unit and "AHU" – Archive Handling Unit.
mac_address	varchar(20)	null	The MAC address coded into the network card used for communication by the NGAS Host. This is an address like: "05:4E:14:8A:11:2B".
n_slots	tinyint	null	Number of slots in the NGAS Node.
cluster_name	varchar(10)	null	Name of the NGAS Cluster this system belongs to.
installation_date	datetime	null	Date the OS and NG/AMS running on the NGAS Host have been installed.
idate	datetime	null	Date this row was inserted.
srv_version	varchar(20)	null	Version of the NG/AMS Server.
srv_port	int	null	Port used by the NG/AMS Server.
srv_archive	tinyint	null	Indicates if the NG/AMS Server is configured to allow Archive Requests (1 = accept Archive Requests).
srv_retrieve	tinyint	null	Indicates if the NG/AMS Server is configured to allow Retrieve Requests (1 = accept Retrieve Requests).
srv_process	tinyint	null	Indicates if the NG/AMS Server is configured to allow

[5] All columns initiated with "srv_" are set by NG/AMS.

			Processing Requests (1 = accept Processing Requests).
srv_data_checking	tinyint	null	Indicates if this server is carrying out Data Consistency Checking (see Section 3.9).
srv_state	varchar(20)	null	Indicates the State of the server.
srv_suspended	tinyint	null	Set to 1 if the server is suspended.
srv_req_wake_up_srv	varchar(32)	null	Name of an NG/AMS Server, which is requested to wake up this host/server that has suspended itself.
srv_req_wake_up_time	datetime	null	Time when the host/server would like to be woken up.

Table 15: Contents of the NGAS Hosts DB Table.

The “ngas_subscr_back_log” and “ngas_subscribers” tables are used by NG/AMS to keep track of the status in connection with Subscribers and of data that could not be delivered. These tables are maintained entirely by NG/AMS, and usually the user does not have to be concerned with these.

8.5 *EXPERT*: Table - " ngas_subscr_back_log "

Column	*Type*	*Value*	*Description*
host_id	varchar(32)	not null	The ID of the host where the Data Provider NG/AMS Server is running.
srv_port	int	not null	The port number used by the Data Provider NG/AMS Server.
subscr_id	varchar(255)	not null	The ID of the Subscriber.
subscr_url	varchar(255)	not null	The Delivery URL submitted by the Subscriber. The Data Provider will POST the data on this URL.
file_id	varchar(64)	not null	NGAS ID of file that could not be delivered.
file_name	varchar(255)	not null	Complete filename of file that could not be delivered.
file_version	int	not null	Version of file that could not be delivered.
ingestion_date	datetime	not null	Date the file was ingested into NGAS.
format	varchar(32)	not null	The format (mime-type) of the file.

Table 16: Contents of the NGAS Subscription Back-Log DB Table.

8.6 *EXPERT*: Table - " ngas_subscribers "

Column	*Type*	*Value*	*Description*
host_id	varchar(32)	not null	The ID of the host where the Data Provider NG/AMS Server is running.
srv_port	int	not null	The port number used by the Data Provider NG/AMS Server.
subscr_prio	tinyint	not null	The priority of the Subscriber as indicated by the Subscriber itself. The priority indicates how fast the data will be delivered to this Subscriber, i.e., how much CPU is allocated to deliver the file.
subscr_id	varchar(255)	not null	See “ngas_subscr_back_log.susbcr_id”.
subscr_url	varchar(255)	not null	See “ngas_subscr_back_log.subscr_url.
subscr_start_date	datetime	null	Date from which data should be considered for delivery for that Subscriber.
subscr_filter_plugin	varchar(64)	null	A Filter Plug-In (see Chapter 21), which will be applied on the data to determine whether to deliver it or not to a Subscriber.
subscr_filter_plugin_pars	varchar(128)	null	Plug-In Parameters to hand over to the Filter Plug-In.
last_file_ingestion_date	datetime	null	The Ingestion Date of the last file delivered to the Subscriber. Used to avoid delivering the same data files in multiple copies to the same Subscriber.

Table 17: Contents of the NGAS Subscribers DB Table.

8.7 *EXPERT*: Synchronizing Distributed NGAS DBs

Operating an NGAS system, which makes use of several, independent DBMS' constitutes a delicate problem in order to ensure that all sites are kept up to date, and to avoid corruption of the data holdings due to 'unforeseen/unwanted' replication between the sites.

NG/AMS does not implement any means whatsoever to synchronize distributed DBs, nor to prevent the various NGAS DB sites from being corrupted. It has been decided to keep NG/AMS completely decoupled from any DBMS specific feature, and it is entirely up to the designers of the NGAS infrastructure to ensure data consistency in the various DB holdings.

As an example of this, it can be mentioned that if disks are prepared in the Archive Facility Site and distributed to various, remote Data Production Sites, it is desirable that 1) The DB at the Archive Facility Site is kept up to date, 2) The NGAS installation at the Production Site recognizes that the disk is already registered as an NGAS Storage Media (see also Section 2.7) and 3) At the same time, after a Storage Media has been completed, has left the Production Site, and is located in an NGAS Host at the Archive Facility Site, that it is no changes introduced to the record for that disk at the remote site, are replicated to the Archive Facility Site NGAS DB.

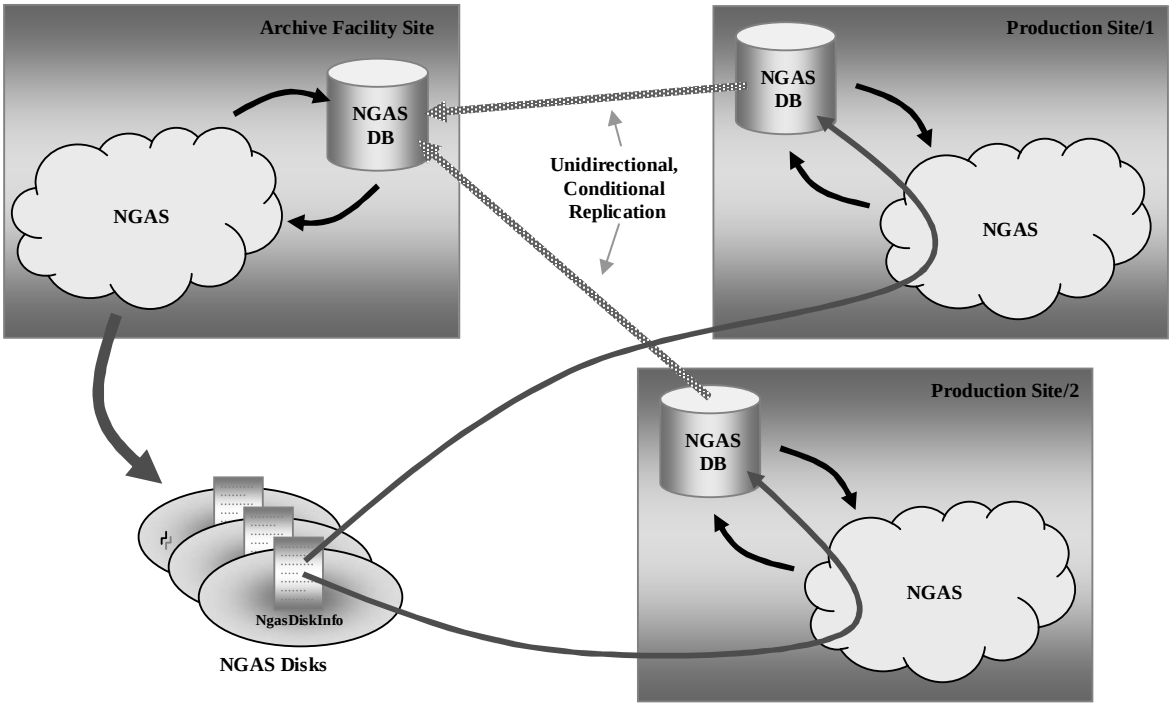

Figure 29: Example of a Distributed NGAS installation using unidirectional, conditional DB replication.

A way to implement this, is as follows[6]:

When NG/AMS registers a new disk, it creates an entry for this in the NGAS DB at the Archive Facility Site. NG/AMS also generates an "NgasDiskInfo" document (see Section 22.2) on the disk. In this way the disk is 'marked' as a 'known' NGAS Disk, and wherever it appears, NG/AMS will recognize the disk and take the information in the "NgasDiskInfo" document and write this to the NGAS DB connected (if the disk is not already registered in that DB). This means that after having prepared the disk at the Archive Facility Site and after this has been received and installed at the Production Site, the disk has now been registered in the NGAS DB at the Production Site, based on the "NgasDiskInfo" document. The Production Site NGAS System, now archives data on the disk. During this phase, all changes in connection with the disk are replicated from the Production Site NGAS DB to the Archive Facility Site NGAS DB (disk information + information about new files). When the disk is completed, it will be marked as such by NG/AMS at the Production Site, and this information replicated to the Archive Facility Site. The disk is subsequently send to the Archive Facility Site where NG/AMS recognizes the disk and updates the information about the disk in the DB at that site. In order to prevent that changes introduced from this

[6] The scenario described here explains how the NGAS DB replication is handled in connection with the ESO NGAS System.

point on, at the Production Site where the disk was residing, are replicated to the Archive Facility Site DB, the DB replication could implement some conditions indicating when to actually update a record in the Archive Facility DB. This could e.g. be done by using the information in the "ngas_hosts" table. If the disk for which information is received for update is located in a host at the Archive Facility Site, such an update is discarded. The replication engine can determine where the disk is located from the columns "ngas_hosts.host_id" and "ngas_hosts.domain". The complete association to do implement is as follows: "ngas_disks.host_id" → "ngas_hosts.host_id" → "ngas_hosts.domain".

Note that using *bi-directional* DB replication may not be an optimal solution either, as unforeseen/unwanted changes in the Production Site DBs are propagated to the Archive Facility Site(s). Using bi-directional replication, it would still be necessary to implement a conditional replication as explained previously.

The scenario described in this section should be seen mainly as an example, and could be used as 'inspiration' when designing the DB system for a distributed NGAS installation.

9 *EXPERT:* The C-API

Together with the NG/AMS package, an API to be used for interfacing C applications with the NG/AMS Server is provided. This is provided in the form of a small library with a set of functions making it easy to communicate from client applications to the NG/AMS Server. Also a number of various macros are provided by the C-API.

The source and the header files for the C-API is contained in the module: "ngams/ngamsCClient". This CVS module contains the files:

File	*Description*
Makefile	Make that generates/compiles the C-API. Can be invoked with the parameters clean and all, e.g.: "make clean all".
ngams.h	Header file for the NG/AMS C-API module. This contains the definition of the function prototypes, and the definition of various macros that can be used in the clients built using the NG/AMS C-API.
ngamsCClient.c	The source file for the NG/AMS C based command line utility. See also Section 5.2 for more information about this tool.
ngamsCClientLib.c	The source file for the library functions provided by the NG/AMS C-API.

Table 18: Source files in the C-API module.

Compiling the "ngamsCClient" module, the following header files and binaries are generated:

File	*Description*
libngams.a	The library to be linked with applications using the NG/AMS C-API.
ngamsCClient	The binary/executable utility, which can be used to communicate with the NG/AMS Server from the command line. Refer to Section 5.2 for further information.
ngamsLICENSE.h	Header file containing the text of the license agreement for NG/AMS (see Chapter 26).
ngamsMAN_PAGE.h	Header file containing the text of the man-page for the NG/AMS C-API command line utility (see Section 9.2).
ngamsVERSION.h	Header file containing the version information for the given distribution of NG/AMS.

Table 19: Files generated compiling the C-API.

In the following sections the header file for the NG/AMS C-API is listed. In addition the man-page for the C-API library functions is shown.

9.1 *EXPERT:* NG/AMS C-API - Header File: “ngams.h”

The source of the NG/AMS C-API header file can be found in the NG/AMS module as follows: "ngams/ngamsCClient/ngams.h". It contains the prototype definitions for the various functions provided by the API, and also the definition of various macros.

9.2 *EXPERT:* NG/AMS C-API - Man Page

The man-page for the NG/AMS C-API contains the following information:

```
NAME
ngamsArchive(), ngamsArchiveFromMem(), ngamsClone(), ngamsCmd2No(),
ngamsCmd2Str(), ngamsDumpErrStdout(), ngamsEncodeUrlVal(), ngamsExit(),
ngamsFreeStatus(), ngamsInitStatus(), ngamsIsDir(), ngamsLabel(),
ngamsLicense(), ngamsOffline(), ngamsOnline(), ngamsRegister(),
ngamsRemDisk(), ngamsRemFile(), ngamsRetrieve2File(), ngamsStat2Str(),
ngamsStatus(), ngamsSubscribe(), ngamsToUpper(), ngamsUnsubscribe(),
ngamsVersion() - C functions to interface to NG/AMS


SYNOPSIS
#include "ngams.h"

In general for the NG/AMS interface functions listed below,
the "host" parameter is the name of the host where the NG/AMS Server
is running. E.g.: "arcdev1.hq.eso.org". The "port" parameter is the
socket port, which the NG/AMS Server is waiting on.
```

If the parameter "wait" is set to 0, an immediate reply to the request will be generated, i.e. before the request has been handled.

The parameter "status" is a structure containing the following members:

Data Type	Member	Description
ngamsSMALL_BUF	date	Date for handling query.
ngamsSTAT	errorCode	Error code giving status for the query. See #1.
ngamsSMALL_BUF	hostId	Host ID for host where the NG/AMS Server is running.
ngamsHUGE_BUF	message	Message from the NG/AMS Server.
ngamsSMALL_BUF	status	Status of query ("OK" | "FAILURE").
ngamsSMALL_BUF	state	State of the NG/AMS Server.
ngamsSMALL_BUF	subState	Sub-State of the NG/AMS Server.
ngamsSMALL_BUF	version	Version of the NG/AMS Server.
char	replyData	Pointer array of pointers pointing to allocated buffers containing the reply data from the request.

#1: The following error codes (internal to the NG/AMS C API) are defined (data type: ngamsSTAT):

Error Macro	Description
ngamsSTAT_SUCCESS	Query successfully executed.
ngamsERR_HOST	No such host.
ngamsERR_SOCK	Cannot create socket.
ngamsERR_CON	Cannot connect to host/server.
ngamsERR_COM	Problem occurred during socket connection.
ngamsERR_TIMEOUT	Timeout encountered while communication with server.
ngamsERR_WR_HD	Write error on socket while writing header.
ngamsERR_WR_DATA	Write error on socket while writing data.
ngamsERR_RD_DATA	Read error while reading data.
ngamsERR_INV_REPLY	Invalid answer from data server.
ngamsERR_FILE	Invalid filename specified.
ngamsERR_ALLOC_MEM	Cannot allocate memory.
ngamsERR_UNKNOWN_STAT	Unknown status code.
ngamsERR_UNKNOWN_CMD	Unknown command issued.
ngamsERR_INV_TARG_FILE	Invalid target filename specified.
ngamsERR_INV_PARS	Invalid parameters given.
ngamsSRV_OK	Request successfully handled by server.
ngamsSRV_REDIRECT	The reply is an HTTP redirection response.
ngamsSRV_INV_QUERY	Invalid query.

Apart from that, the errors defined by NG/AMS can be returned.

All functions return ngamsSTAT_SUCCESS in case of success. In case of an error a termination status within the set of status codes given above.

The following macros are defined for referring to NG/AMS commands:

Command Macros (#2)	Description
ngamsCMD_ARCHIVE	Archive file.
ngamsCMD_CLONE	Clone files.
ngamsCMD_EXIT	Make NG/AMS Server exit.
ngamsCMD_LABEL	Make NG/AMS print out a disk label.
ngamsCMD_ONLINE	Bring NG/AMS Server Online.

```
ngamsCMD_OFFLINE          Bring NG/AMS Server Offline.
ngamsCMD_REGISTER         Register files on a disk.
ngamsCMD_REMDISK          Remove a disk from NGAS.
ngamsCMD_REMFILE          Remove a file from NGAS.
ngamsCMD_RETRIEVE         Retrieve a file.
ngamsCMD_STATUS           Query status information from NG/AMS.
ngamsCMD_SUBSCRIBE        Subscribe to a NG/AMS Server.
ngamsCMD_UNSUBSCRIBE      Un-subscribe/cancel a previous subscription.

#2: All command macros exist also as a string, which carries the name
    of the enumerated macro name with a "_STR" appended.


In the following, the functions provided for interacting with NG/AMS
are listed. The specific parameters for each function are listed in
connection with the function. The parameters used by several
functions are as follows:

host:         Host name of NG/AMS Server to contact.

port:         Port number used by NG/AMS Server to contact.

timeoutSecs:  Timeout in seconds to apply while executing the request.

wait:         Wait for the NG/AMS Server to finish the request (=1)
              completely, or return an immediate response (=0).

status:       Pointer to ngamsSTATUS structure containing the
              status of the query.

The functions provided by the NG/AMS C-API:

ngamsSTAT ngamsArchive(const char          *host,
                       const int           port,
                       const float         timeoutSecs,
                       const char          *fileUri,
                       const char          *mimeType,
                       const int           noVersioning,
                       const int           wait,
                       ngamsSTATUS         *status)
Archive a file into NGAS.

fileUri:      Reference name for the file to archive.

mimeType:     The mime-type of the file to archive. In some cases
              it is not possible for NG/AMS to determine the mime-type
              of a data file to be archived, e.g. when the file being is
              archived is RETRIEVEd from another NGAS Host. For efficiency
              it is thus better to indicate the mime-type to enable
              NG/AMS to store the file directly on the target disk.
              If not use this can be put to "".

noVersioning: If set to 1 no new File Version will be generated for
              the file being archived even though a file with that
              File ID was already existing.


ngamsSTAT ngamsArchiveFromMem(const char          *host,
                              const int           port,
                              const float         timeoutSecs,
                              const char          *fileUri,
                              const char          *bufPtr,
                              const int           size,
                              const char          *mimeType,
                              const int           noVersioning,
                              const int           wait,
                              ngamsSTATUS         *status)
Archive a file which contents have been loaded into a buffer in memory.

fileUri,
mimeType,
noVersioning: See ngamsArchive().

bufPtr:       Pointer to buffer containing the contents of the file.
```

```
size:         Size in bytes of the data loaded into memory.


ngamsSTAT ngamsClone(const char          *host,
                     const int           port,
                     const float         timeoutSecs,
                     const char          *fileId,
                     const int           fileVersion,
                     const char          *diskId,
                     const int           wait,
                     ngamsSTATUS         *status)
Execute a CLONE command. For the exact interpretation of various
combinations of fileId, fileVersion and diskId, consult the man-page
for the NG/AMS Python module "ngamsCloneCmd", function: "clone()".

fileId:          ID of file to clone.

fileVersion:     Version of files to be taken into account for the
                 cloning.

diskId:          Disk ID for the files to be taken into account.


ngamsSTAT ngamsCmd2No(const ngamsSMALL_BUF    cmdStr,
                      ngamsCMD                *cmdCode)
Convert a command given as string into the corresponding code (integer).

cmdStr:    Command name as string.

cmdCode:   Command code as defined by the enumerated type ngamsCMD.


ngamsSTAT ngamsCmd2Str(const ngamsCMD    cmdCode,
                       ngamsSMALL_BUF    cmdStr)
Convert an NG/AMS command given as a code (integer) to a string.

cmdCode,
cmdStr:    See ngamsCmd2No().


void ngamsDumpErrStdout(const ngamsSTATUS  *status)
Dump an error message in the status structure on stdout.

status:   Pointing to instance of the ngamsSTATUS structure containing
          the information to print out.


char *ngamsEncodeUrlVal(const char *urlVal,
                        const int  skipScheme)
Encode the value given as input parameter to replace special
characters to make the value suitable for usage in a URL.

urlVal:            Value to be encoded.

skipScheme:        If the value is initiated with an HTTP scheme
                   (ftp:, http:, file:), this will not be encoded
                   if this flag is set to 1.


void ngamsFreeStatus(ngamsSTATUS         *status)
Free the memory occupied by the status object.

status:   Pointing to instance of the ngamsSTATUS structure containing
          the information to print out.


void ngamsInitStatus(ngamsSTATUS         *status)
Initialize the ngamsSTATUS structure, making it ready for usage within
the NG/AMS C-API functions.

status:   Pointing to instance of the ngamsSTATUS structure containing
          the information to print out.
```

```
int ngamsIsDir(const char  *filename)
Return 1 if the filename given is a directory, otherwise 0 is returned.

filename:    Name of directory to probe.


ngamsSTAT ngamsExit(const char          *host,
                    const int           port,
                    const float         timeoutSecs,
                    const int           wait,
                    ngamsSTATUS         *status)
Send an EXIT command to the NG/AMS Server to make it
clean up and terminate execution.


ngamsSTAT ngamsLabel(const char          *host,
                     const int           port,
                     const float         timeoutSecs,
                     const char          *slotId,
                     ngamsSTATUS         *status)
Send a LABEL command to the NG/AMS Server.

slotId:    ID of slot hosting the disk for which to generate the label.


char *ngamsLicense(void)
Return pointer to buffer containing the NG/AMS License Agreement.


ngamsSTAT ngamsOffline(const char          *host,
                       const int           port,
                       const float         timeoutSecs,
                       const int           force,
                       const int           wait,
                       ngamsSTATUS         *status)
Send an OFFLINE command to the NG/AMS Server to bring it to Offline State.

force:     Force the server to go Offline immediately, even though it is
           busy.


ngamsSTAT ngamsOnline(const char          *host,
                      const int           port,
                      const float         timeoutSecs,
                      const int           wait,
                      ngamsSTATUS         *status)
Send an ONLINE command to the NG/AMS Server to bring it to Online State.


ngamsSTAT ngamsRegister(const char          *host,
                        const int           port,
                        const float         timeoutSecs,
                        const char          *path,
                        const int           wait,
                        ngamsSTATUS         *status)
Send an REGISTER command to the NG/AMS Server to make it register
a file or set of files.

path:      Path starting point from which the scanning for files to
           register will be initiated. Only files with a known mime-type
           is taken into account.


ngamsSTAT ngamsRemDisk(const char        *host,
                       const int         port,
                       const float       timeoutSecs,
                       const char        *diskId,
                       const int         execute,
                       ngamsSTATUS       *status)
Send a REMDISK command to the NG/AMS Server. If execute is 0 the
disk information will not be deleted from the NGAS DB and from the
the disk itself. Otherwise, if 1 is specified, this information will
will be deleted.
```

```
diskId:      ID of disk to remove.

execute:     If set to 1 the command will be executed and the disk
             removed from the system (if possible). Otherwise a report
             will be send back indicating if it is possible to
             remove the disk referenced.


ngamsSTAT ngamsRemFile(const char        *host,
                       const int         port,
                       const float       timeoutSecs,
                       const char        *diskId,
                       const char        *fileId,
                       const int         fileVersion,
                       const int         execute,
                       ngamsSTATUS       *status)
Send a REMFILE command to the NG/AMS Server. If execute is 0 the
disk information will not be deleted from the NGAS DB and from the
the disk itself. Otherwise, if 1 is specified, this information will
will be deleted. For the interpretation of various combinations of
the parameters diskId, fileId, fileVersion and execute consult the
man-page of the Python module "ngamsRemoveCmd", function remFile().

diskId:        ID of disk hosting the file(s) to be removed.

fileId:        ID of file(s) to be removed.

fileVersion:   Version of file(s) to be removed.

execute:       If set to 1 the files will be removed (if possible),
               otherwise a report will be send back indicating what
               would be removed if the command is executed.


ngamsSTAT ngamsRetrieve2File(const char          *host,
                             const int           port,
                             const float         timeoutSecs,
                             const char          *fileId,
                             const int           fileVersion,
                             const char          *processing,
                             const char          *processingPars,
                             const char          *targetFile,
                             ngamsSTATUS         *status);
Send a RETRIEVE command to the NG/AMS Server to retrieve a
data file, and store this in a file on the local disk.

fileId:           ID of the file to retrieve.

fileVersion:      Specific version of file to retrieve. If set to -1 the
                  latest version will be retrieved.

processing:       Name of DPPI to be invoked by NG/AMS when data is
                  retrieved.

processingPars:   Optional list of parameters to hand over to the DPPI.

targetFile:       If a valid filename is specified the data retrieved
                  will be stored in a file with that name. If a directory
                  is given, the data file retrieved will be stored in that
                  directory with the name under which it is stored in
                  NGAS. If this parameter is an empty string, it will be
                  tried to stored the file retrieved under the NGAS
                  archive name in the current working directory.


ngamsSTAT ngamsStat2Str(const ngamsSTAT  statNo,
                        ngamsMED_BUF     statStr)
Convert a status code (ngamsSTAT) to a readable string.

statNo:    Status as code.

statStr:   Status as string.
```

```
ngamsSTAT ngamsStatus(const char          *host,
                      const int           port,
                      const float         timeoutSecs,
                      ngamsSTATUS         *status)
Send a STATUS command to NG/AMS to query the current status of the
system. No parameters are defined at present.


ngamsSTAT ngamsSubscribe(const char          *host,
                         const int           port,
                         const float         timeoutSecs,
                         const char          *url,
                         const int           priority,
                         const char          *startDate,
                         const char          *filterPlugIn,
                         const char          *filterPlugInPars,
                         ngamsSTATUS         *status)
Send a SUBSCRIBE to NG/AMS to subscribe for data or a specific type
of data.

url:                 Subscriber URL to where data is pushed.

priority:            Priority of the Subscriber (low number = high
                     priority). Default value 10.

startDate:           Start date defining which data the subscription
                     should take into account.

filterPlugIn:        Optional Filter Plug-In to apply when selecting
                     which data files to deliver to the clients.

filterPlugInPars:    Optional Filter Plug-In Parameters to transfer
                     to the Filter Plug-In.


void ngamsToUpper(char *str)
Convert a string to upper characters.

str:   Pointer to string to convert.


char *ngamsVersion(void)
Return pointer to buffer with the NG/AMS version information.


ngamsSTAT ngamsUnsubscribe(const char          *host,
                           const int           port,
                           const float         timeoutSecs,
                           const char          *url,
                           ngamsSTATUS         *status)
Send an UNSUBSCRIBE to NG/AMS to subscribe for data or a specific type
of data.

url:                 Subscriber URL to where data is pushed.



CAUTIONS
This is a first implementation of the module. Changes may be
introduced in order to improve the usability of the API.

Remember to perform a call to ngamsInitStatus() right after declaring
this, and a call to ngamsFreeStatus() after each call to one of the
commands used to issue commands to NG/AMS. Memory may be allocated
dynamically and needs to be freed.


EXAMPLES
To archive a file using the API the following must be called from
the application:

#include "ngams.h"
```

```
ngamsSTATUS      status;
if (ngamsArchive("wfinau", "7171", 10, "/home/data/MyFile.fits", "",
                 1, 0, &status) != ngamsSTAT_SUCCESS)
    {
    ngamsDumpErrStdout(&status);
    ... error handling ...
    }


To retrieve a file into the directory "/home/data/target_dir". The
name will be the same as the File ID:

#include "ngams.h"

ngamsSTATUS      status;
if (ngamsRetrieve2File("wfinau", "7171", 30,
                       "WFI.2001-10-21T23:24:03.925",
                       -1, "", "", "/home/data/target_dir",
                       &status) != ngamsSTAT_SUCCESS)
    {
    ngamsDumpErrStdout(&status);
    ... error handling ...
    }



CAUTIONS
If data is returned by the NG/AMS Server the member "replyData" will
have a pointer pointing to each block of data. It is the responsibility
of the calling application to free these data chunks. The function
"ngamsFreeStatus()" can be used for this.




- - - - - -
Last change:  24/12/02-12:24
```

Figure 30: The functions and macros of the NG/AMS C-API.

10 *EXPERT:* The Python API

The NG/AMS Python API can be used by Python applications to interface with the NG/AMS Server in an easy and straightforward manner. The API hides most of the technical details of the NG/AMS communication interface.

To use the Python API, the following "import" statements must be contained in the client application:

```
from   ngams import *
import ngamsPClient

…
```

Figure 31: Using the NG/AMS Python-API.

The API provides a class "ngamsPClient", which is contained in the Python module "ngamsPClient.py". The complete documentation for the API is contained as in-line, Python documentation strings in the source file. It is suggested to browse this documentation online using a WEB browser and the "pydoc" utility. See also Section 23.2. Here only a summary of the API is given. The most significant methods provided by this class are:

Method	*Description*
archive()	Archive a file to NG/AMS. This can be either done by the Archive Pull Technique or by the Archive Push Technique. The technique applied depends on the File URI given.
clone()	Send a CLONE command to NG/AMS.
exit()	Used to issue an EXIT command to a running NG/AMS Server.
getHost()/setHost()	Get/set the NGAS Host reference indicating where the NG/AMS Server with which there is communicated is running.
getPort()/setPort()	Get/set the NGAS port reference indicating the port used by the NG/AMS Server with which there is communicated.
handleCmd()	Execute a command from a set of command line parameters; mostly relevant for the NG/AMS Python based command line utility based on the Python API.
init()	Send an INIT command to the NG/AMS Server.
label()	Send a LABEL command to the NG/AMS Server.
offline()	Send a OFFLINE command to the NG/AMS Server.
online()	Send a ONLINE command to the NG/AMS Server.
pushFile()	Carry out the necessary actions for performing an Archive Push Request.
register()	Send a REGISTER command to NG/AMS to register files stored in a given location.
remDisk()	Send a REMDISK command to NG/AMS to remove a disk from the system.
remFile()	Send a REMFILE command to NG/AMS to remove a file or set of files from the system.
retrieve2File()	Retrieve a file into a file on the local disk.
sendCmd()/sendCmdGen()	Handle the sending of a command and reception of the reply to this command.
status()	Send a STATUS command to the NG/AMS Server.
subscribe()	Send a SUBSCRIBE command to the NG/AMS Server to subscribe as client to a certain kind of data.
unsubscribe()	Send an UNSUBSCRIBE command to the NG/AMS Server to unsubscribe a client.

Table 20: Methods/functions in the Python-API.

A small example application based on the NG/AMS Python API is listed in the following. It is used to archive a file:

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsPClientEx.py,v 1.2 2002/02/26 17:25:41 safcvs Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------
# jknudstr  26/02/2002  Created
#
"""
Small example application archiving a file.
"""

import sys
from   ngams import *
import ngamsPClient

# Check the input parameters.
if (len(sys.argv) != 4):
    print "Correct usage is:\n"
    print "ngamsPClientEx <host> <port> <file URI>\n"
    sys.exit(1)

# Get the parameters for handling the archiving.
host    = sys.argv[1]
port    = sys.argv[2]
fileUri = sys.argv[3]

# Create instance of NG/AMS Python API.
client = ngamsPClient.ngamsPClient(host, port)

# Execute the command.
status = client.archive(fileUri)

# Handle result - here we simply print the XML status message to stdout.
print status.genXml(0, 1, 1, 1).toprettyxml('  ', '\n')[0:-1]

#
# ___oOo___
```

Figure 32: Small example program using the Python-API (FILE: "ngams/ngamsPClient/ngamsPClientEx").

This small test program will generate an output as the following on stdout while archiving the file (example):

```
ngasmgr@acngast1:/opsw/NGAS/ngams> python ngamsPClient/ngamsPClientEx.py acngast1 7777
/home/ngasmgr/tmp/WFI.2001-09-15T22\:49\:07.652.fits
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-12-31T09:28:09.251" HostId="acngast1" Message="Successfully handled Archive Push Request
          for data file with URI: WFI.2001-09-15T22:49:07.652.fits" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
  <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32300" BytesStored="8709834319" Checksum="" Completed="0"
              CompletionDate="" DiskId="IC35L040AVER07-0-SXPTX093675" HostId="acngast1"
              InstallationDate="2002-11-25T09:48:25.000" LastCheck="" LogicalName="FITS-M-000001"
              Manufacturer="IBM" MountPoint="/NGAS/data1" Mounted="1" NumberOfFiles="163" SlotId="1"
              TotalDiskWriteTime="905.324898006" Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1810827525" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="WFI.2001-09-15T22:49:07.652"
                FileName="saf/2001-09-15/1/WFI.2001-09-15T22:49:07.652.fits.Z" FileSize="142074506"
                FileStatus="00000000" FileVersion="1" Format="application/x-cfits" Ignore="0"
                IngestionDate="2002-12-31T09:28:08.000" Tag="" UncompressedFileSize="141546240"/>
  </DiskStatus>
</NgamsStatus>
ngasmgr@acngast1:/opsw/NGAS/ngams>
```

Figure 33: Output on "stdout" from example program using the Python-API.

11 *EXPERT:* The NG/AMS Plug-In API

The NG/AMS Plug-In API provides convenience functions to facilitate the implementation of the various types of plug-ins used within the context of NG/AMS. The actual thorough documentation is contained as inline Python documentation strings in the code itself. For further information about this issue consult Section 23.2.

It is recommended to restrict the usage of functions from NG/AMS modules to only the one ones contained in the NG/AMS Plug-In API (Python Module: "ngamsPlugInApi.py"). It should be mentioned, that for the moment the amount of convenience functions provided is limited. Basically only the functions needed for implementing the plug-ins provided so far, have been considered in this context. If new functions are needed requests for such can be issued to: ngast@eso.org. Here is an overview of the convenience functions provided by the NG/AMS Plug-In API for the moment:

Function	*Description*
determineMimeType()	From the filename of a data file, determine the mime-type of the file.
execCmd()	Execute a command on the shell, and return a tuple with the following information: [<exit code>, <stdout>, <stderr>].
genDapiSuccessStat()	Generate a return status object as it must be returned from a DAPI.
genFileInfo()	Extract the information about a file and return this to the plug-in.
genFileInfoReg()	Extract the file information needed in the context of a Register Plug-In.
genNgasId()	Generate the ID for this NGAS installation.
genRegPiSuccessStat()	Generate return status for a Register Plug-In.
getDppiPars()	Get the parameters specified for a DPPI in the configuration in raw format (as given in the configuration).
getFileSize()	Get the size of a file.
getFitsKeys()	Extract a number of FITS keyword cards from a FITS file.
notify()	Send a Notification Message from the plug-in.
parseDapiPlugInPars()	Get the plug-in parameters. This function is dedicated to be used by DAPIs.
parseRawPlugInPars()	Parse the plug-in parameters. These are supposed to be defined in the following format: "<parameter>=<value>,<parameter>=<value>,…". The parameters and values are returned in a dictionary whereby the keys of the dictionary are the parameter names.
parseRegPlugInPars()	Get the parameters for a Register Plug-In referred to by its mime-type from the configuration, parse these and return them in a dictionary.
prepProcFile()	The function is used to create a copy of a file to be processed in the Processing Directory.

Table 21: Functions in the NG/AMS Plug-In API.

Examples of plug-in can be found in the Chapters: 12 (System Online Plug-In), 13 (System Offline Plug-In), 14 (The Label Printer Plug-In), **15 (The Data Handling Plug-In)** , 17 (The Data Processing Plug-In) and 18 (The Data Checksum Plug-In).

Apart from the functions contained in the module "ngamsPlugInApi.py", the following classes are used for implementing the plug-ins: "ngamsServer", "ngamsConfig", "ngamsReqProps", "ngamsDppiStatus", "ngamsDb", "ngamsPhysDiskInfo" (NG/AMS Disk Dictionary). These classes are all described in more details in Chapter 23.

Frequently needed in plug-in is access to the NG/AMS Configuration and to the NGAS DB. Access to these can be obtained by means of the methods “ngamsServer.getCfg()” and “ngamsServer.getDb()”. A reference to the “ngamsServer” object is handed over to all types of NG/AMS plug-in functions.

To be able to write efficiently, plug-ins for NG/AMS, it is required to have a more a less profound overview of the NG/AMS SW, or at least this will be of major advantage, depending on the complexity of the tasks performed by the plug-ins. An overview of the NG/AMS SW is given in Chapter 23.

12 *EXPERT:* The System Online Plug-In

The purpose of the System Online Plug-In, is to prepare the system for the Online State, where it must be fully operational according to the configuration. During this phase the storage disks are usually mounted and possibly checked for proper functioning and accessibility. A very essential task of a System Online Plug-In is to generate the so-called Physical Disk Dictionary. This contains the 'physical' information about the disks installed in an NGAS Host.

The plug-in is invoked by NG/AMS when it is going Online, i.e., either when it has received an ONLINE command or when it has been started with the "-autoOnline" command line parameter. The actual implementation depends highly on the context (HW) in use and other specific requirements in connection with an NGAS Node. The System Online Plug-In is not executed when running in Simulation Mode (see Section 3.6).

12.1 *EXPERT:* Interface of a System Online Plug-In

The System Online Plug-In must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A System Online Plug-In has an interface as shown in Figure 34.

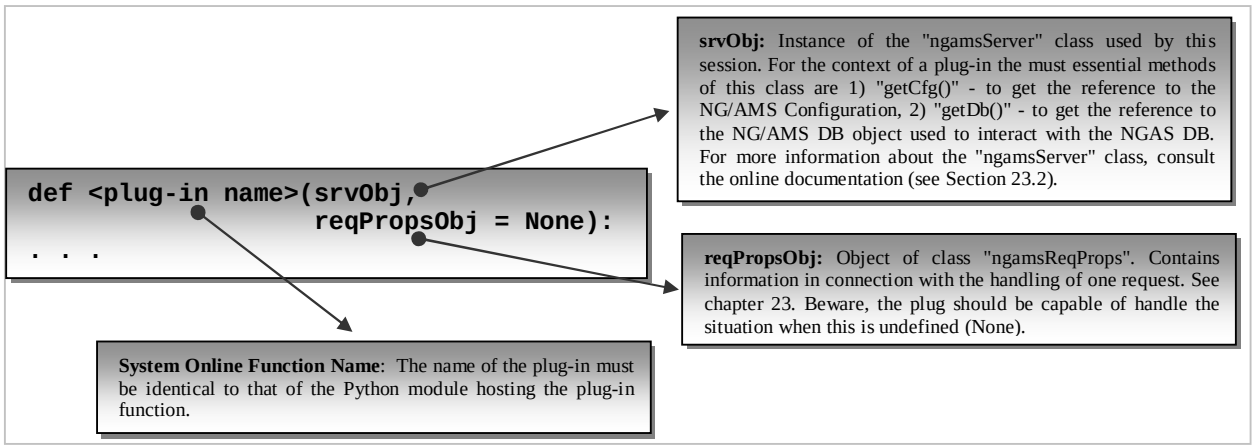

Figure 34: Function interface of a System Online Plug-In.

The return value of a System Online Plug-In is the Physical Disk Dictionary. This must be generated by the plug-in. It is a standard Python dictionary with "ngamsPhysDiskInfo" objects stored in it. The Slot IDs of the disks are used as keys in the dictionary. The Disk Dictionary is very essential for the proper operation of NG/AMS. It is therefore crucial that the plug-in extracts and generates this information correctly for NG/AMS. The contents of the Physical Disk Dictionary is depicted in Figure 35.

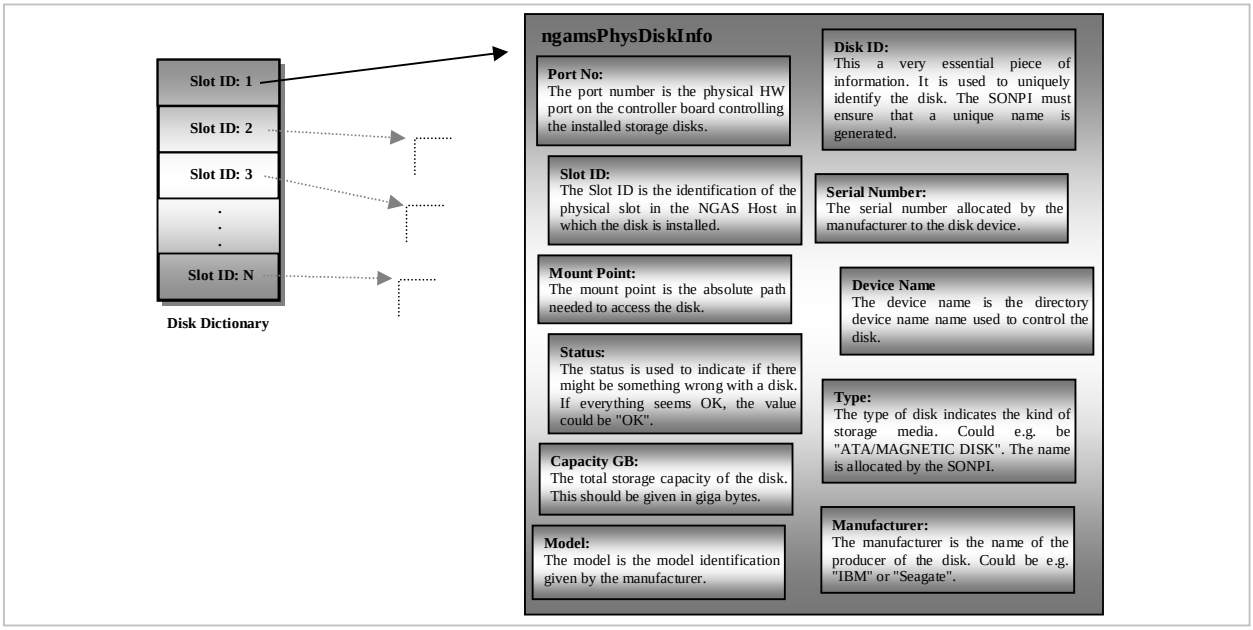

Figure 35: The NG/AMS Physical Disk Dictionary.

An exception must be thrown in case errors occur during the process of bringing the system to Online State.

12.2 *EXPERT:* Example System Online Plug-In

In the following an example System Online Plug-In, which is used for the moment for the NGAS installation for WFI at the La Silla 2.2m telescope. It is perhaps not a very good example of such a plug-in since most of the code is distributed in other modules. Please check the Python source files "ngams/ngamsPlugIns/ngamsEscalada6800Utils.py" and "ngams/ngamsPlugIns/ngamsLinuxSystemPlugInApi.py" for further information.

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsLinuxOnlinePlugIn.py,v 1.16 2002/07/10 08:34:33 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------
# jknudstr  10/05/2001  Created.
#

"""
Module that contains a System Online Plug-In used by the ESO NGAS
installations.
"""


from   ngams import *
import ngamsPlugInApi
import ngamsLinuxSystemPlugInApi, ngamsEscalada6800Utils


def ngamsLinuxOnlinePlugIn(srvObj,
                           reqPropsObj = None):
    """
    Function mounts all NGAMS disks and loads the kernel module for the IDE
    controller card. It returns the NGAMS specific disk info dictionary.

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    Returns:       Disk info dictionary (dictionary).
    """
    rootMtPr = srvObj.getCfg().getMountRootDirectory()
    parDic = ngamsPlugInApi.\
             parseRawPlugInPars(srvObj.getCfg().getOnlinePlugInPars())
    stat = ngamsLinuxSystemPlugInApi.insMod(parDic["module"])
    if (stat == 0):
```

```
        msg = "Kernel module " + parDic["module"] + " loaded"
        info(1, msg)
        diskDic = ngamsEscalada6800Utils.parseHtmlInfo(parDic["uri"], rootMtPr)
        ngamsLinuxSystemPlugInApi.removeFstabEntries(diskDic)
        ngamsLinuxSystemPlugInApi.ngamsMount(srvObj.getDb(), diskDic)
        return diskDic
    else:
        errMsg = "Problem executing ngamsLinuxOnlinePlugIn"
        errMsg = genLog("NGAMS_ER_ONLINE_PLUGIN", [errMsg])
        error(errMsg)
        raise exceptions.Exception, errMsg


if __name__ == '__main__':
    """
    Main function.
    """
    import sys
    import ngamsConfig, ngamsDb

    setLogCond(0, "", 0, "", 1)

    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% python ngamsLinuxOnlinePlugIn <NGAMS cfg>\n"
        sys.exit(0)

    ngamsCfgObj = ngamsConfig.ngamsConfig()
    ngamsCfgObj.load(sys.argv[1])
    dbConObj = ngamsDb.ngamsDb(ngamsCfgObj.getDbServer(),
                               ngamsCfgObj.getDbName(),
                               ngamsCfgObj.getDbUser(),
                               ngamsCfgObj.getDbPassword())
    dbConObj.query("use " + ngamsCfgObj.getDbName())
    diskDic = ngamsLinuxOnlinePlugIn(dbConObj, ngamsCfgObj)
    print "Disk Dictionary = ", str(diskDic)


# --- oOo ---
```

Figure 36: Example System Online Plug-In (FILE: “ngams/ngamsPlugIns/ngamsLinuxOnlinePlugIn.py”.

13 *EXPERT:* The System Offline Plug-In

The purpose of the System Offline Plug-In, is to prepare the system for the Offline State, where it should be put to its 'standby condition'. During this procedure, the disks could be unmounted and other actions performed like e.g. unloading of SW modules used for accessing the Storage Media.

13.1 *EXPERT:* Interface of a System Offline Plug-In

The function interface of a System Offline Plug-In is the same as for the System Online Plug-In (see 12.1). A System Offline Plug-In does not return any information to NG/AMS. An exception must be thrown in case errors occur during the process of bringing the system to the Offline State.

13.2 *EXPERT:* Example System Offline Plug-In

In the following an example System Offline Plug-In, which is used for the moment for the NGAS installation for WFI at the La Silla 2.2m telescope. It is perhaps not a very good example of such a plug-in since most of the code is distributed in other modules. Check the Python source files "ngams/ngamsPlugIns/ngamsEscalada6800Utils.py" and "ngams/ngamsPlugIns/ngamsLinuxSystemPlugInApi.py" for further information.

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsLinuxOfflinePlugIn.py,v 1.10 2002/07/10 08:34:33 arcsw Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  10/05/2001  Created.
#

"""
Module that contains a System Offline Plug-In used by the ESO NGAS
installations.
"""

from   ngams import *
import ngamsPlugInApi
import ngamsLinuxSystemPlugInApi, ngamsEscalada6800Utils


def ngamsLinuxOfflinePlugIn(srvObj,
                            reqPropsObj = None):
    """
    Function unmounts all NGAMS disks and removes the kernel module for
    the IDE controller card.

    srvObj:        Reference to instance of the NG/AMS Server class
                   (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    Returns:       Void.
    """
    rootMtPr = srvObj.getCfg().getMountRootDirectory()
    parDicOnline = ngamsPlugInApi.\
                   parseRawPlugInPars(srvObj.getCfg().getOnlinePlugInPars())
    diskDic = ngamsEscalada6800Utils.parseHtmlInfo(parDicOnline["uri"],
                                                   rootMtPr)
    parDicOffline = ngamsPlugInApi.\
                    parseRawPlugInPars(srvObj.getCfg().getOfflinePlugInPars())

    # This is only unmounting the NGAMS disks and may lead to problems
    # if someone mounts other disks off-line.
    if (parDicOffline.has_key("unmount")):
        unmount = int(parDicOffline["unmount"])
    else:
        unmount = 1
    if (unmount):
        ngamsLinuxSystemPlugInApi.ngamsUmount(diskDic)
        stat = ngamsLinuxSystemPlugInApi.rmMod(parDicOnline["module"])
        if (stat):
            errMsg = "Problem executing ngamsLinuxOfflinePlugIn! " +\
                     "The system is in not in a safe state!"
            errMsg = genLog("NGAMS_ER_OFFLINE_PLUGIN", [errMsg])
            error(errMsg)
            raise exceptions.Exception, errMsg
        msg = "Kernel module " + parDicOnline["module"] + " unloaded"
```

```
        info(1,msg)


if __name__ == '__main__':
    """
    Main function.
    """
    import sys
    import ngamsConfig, ngamsDb

    setLogCond(0, "", 0, "", 1)

    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% python ngamsLinuxOfflinePlugIn <NGAMS cfg>\n"
        sys.exit(0)

    ngamsCfgObj = ngamsConfig.ngamsConfig()
    ngamsCfgObj.load(sys.argv[1])
    dbConObj = ngamsDb.ngamsDb(ngamsCfgObj.getDbServer(),
                               ngamsCfgObj.getDbName(),
                               ngamsCfgObj.getDbUser(),
                               ngamsCfgObj.getDbPassword())
    dbConObj.query("use " + ngamsCfgObj.getDbName())
    ngamsLinuxOfflinePlugIn(dbConObj, ngamsCfgObj)


# --- oOo ---
```

Figure 37: Example System Offline Plug-In (FILE: “ngams/ngamsPlugIns/ngamsLinuxOfflinePlugIn.py”).

14 *EXPERT:* The Label Printer Plug-In

The purpose of the Label Printer Plug-In is to print a label on request from NG/AMS on the label printer installed on the NGAS Host. The plug-in must generate the appropriate control sequence of characters in order to request the printer to produce the label. Also other actions needed to control the printer should be taken care of by the plug-in. I.e., the plug-in could be seen as a high-level/intelligent printer driver.

14.1 *EXPERT:* Interface of a Label Printer Plug-In

A Label Printer Plug-In must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Label Printer Plug-In has an interface as shown in Figure 38.

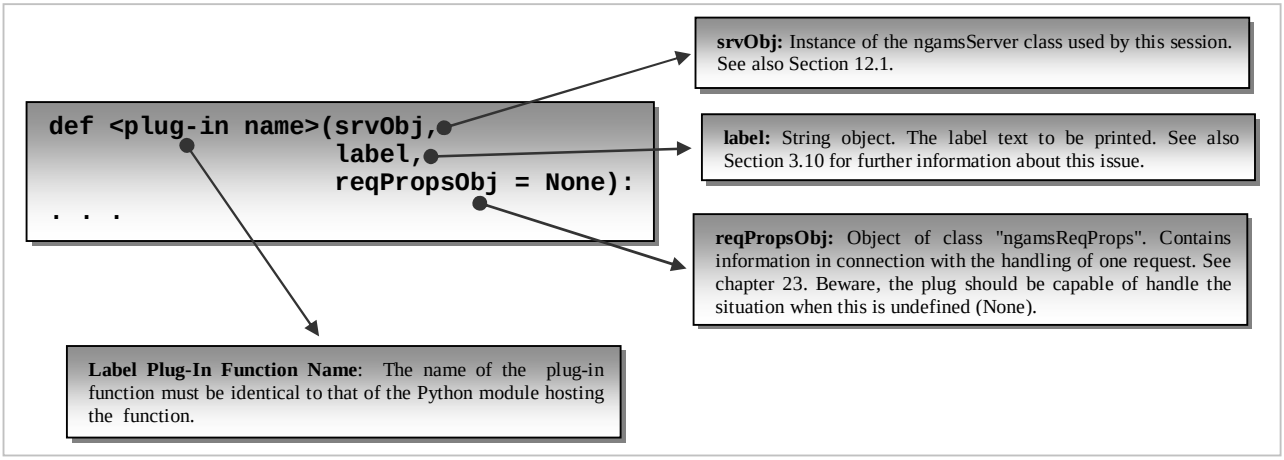

Figure 38: Function interface of a Label Printer Plug-In.

A Label Printer Plug-In does not return any data to NG/AMS. An exception must be thrown in case errors occur during the printing process.

14.2 *EXPERT:* Example of a Label Printer Plug-In

In the following the source code of an example is shown. This is used to control a Brother label printer (Brother P-Touch, 9200 DX).

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsBrotherPT9200DxPlugIn.py,v 1.18 2002/07/10 17:11:41 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# awicenec/
# jknudstr  10/05/2001  Created
#

"""
This module contains a plug-in driver for printing labels on
the Brother PT-9200DX label printer.
"""

import sys, time
from   ngams import *
import ngamsPlugInApi, ngamsConfig


def genFontsDictionary(fnm):

    """
    Function reads the contents of a bitmap character file <fnm>.
    The character contents of this file has to be compliant with the  keys:
```

```
    keys = ['Header','-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
            ':', 'A','B','C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
            'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
            'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
            'z', 'Trailer']

    These keys are used to fill a dictionary with the bitmaps and can
    then be used to print strings on the Brother pTouch 9200DX
    printer.

    Synopsis:  charDict = ngamsGetCharDict(<fnm>)

    fnm:        Filename of font definition file (string).

    Returns:    Return value is a dictionary with the keys
                given above (dictionary).
    """
    keys = ['Header','-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
            ':', 'A','B','C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
            'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
            'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
            'z', 'Trailer']

    try:
        f = open(fnm)
        charArr = f.read()
        f.close()
    except exceptions.Exception, e:
        error(str(e))
        errMsg = "Problems opening CharDict file (" + str(e) + ") "
        raise exceptions.Exception, errMsg

    charArr = charArr.split('ZG')
    charDict = {}
    i = 0
    if len(charArr) != len(keys):
        errMsg = 'Wrong number of characters in CharDict file: ' + fnm
        error(str(e))
        raise exceptions.Exception, errMsg

    for k in keys:
        if k == 'Header' or k == 'Trailer':
            charDict.update({k:charArr[i]})
        else:
            charDict.update({k:'G'+charArr[i]})   # put the G back
        charDict.update({' ':'ZZZZZZZZZZZZZ'})    # add a blank
        i = i + 1

    return charDict

def ngamsBrotherPT9200DxPlugIn(srvObj,
                               label):
    """
    Driver for printing labels on the label printer Brother PT-9200DX.

    srvObj:           Reference to instance of the NG/AMS Server
                      class (ngamsServer).

    label:            Label text to print (string).

    Returns:          Void.
    """
    plugInPars = srvObj.getCfg().getLabelPrinterPlugInPars()
    info(2,"Executing plug-in ngamsBrotherPT9200DxPlugIn with parameters: "+
         plugInPars + " - Label: " + label + " ...")
    parDic = ngamsPlugInApi.parseRawPlugInPars(plugInPars)

    # Get the font bit pattern dictionary.
    fontDic = genFontsDictionary(parDic["font_file"])

    # Generate the printer control code.
    printerCode = fontDic["Header"]
    for i in range(len(label)):
        if (not fontDic.has_key(label[i])):
            errMsg = "No font defintion for character: \"" + label[i] +\
                     "\" - in font definition file: " + parDic["font_file"] +\
                     " - cannot generate disk label: " + label
            error(errMsg)
            ngamsPlugInApi.notify(srvObj.getCfg(), NGAMS_NOTIF_ERROR,
                                  "ngamsBrotherPT9200DxPlugIn: " +\
                                  "ILLEGAL CHARACTER REQ. FOR PRINTING",
                                  errMsg)
```

```
            raise exceptions.Exception, errMsg

        printerCode = printerCode + fontDic[label[i]]
    printerCode = printerCode + fontDic["Trailer"]

    # Generate printer file, write printer control code.
    printerFilename = "/tmp/ngamsLabel_" +\
                      ngamsPlugInApi.genNgasId(srvObj.getCfg()) + ".prn"
    fo = open(printerFilename, "w")
    fo.write(printerCode)
    fo.close()

    # Write the printer code file to the device.
    res = ngamsPlugInApi.execCmd("cat "+printerFilename + " > "+parDic["dev"])
    if (not srvObj._getUnitTest()):
        os.system("rm -f " + printerFilename)
    if (res[0] != 0):
        errMsg = "Problem occurred printing label!"
        error(errMsg)
        ngamsPlugInApi.notify(srvObj.getCfg(), NGAMS_NOTIF_ERROR,
                              "ngamsBrotherPT9200DxPlugIn: " +\
                              "PROBLEM PRINTING LABEL", errMsg)
        raise exceptions.Exception, errMsg

    info(2,"Executed plug-in ngamsBrotherPT9200DxPlugIn with parameters: "+
         plugInPars + " - Label: " + label + " ...")


if __name__ == '__main__':
    """
    Main function.
    """
    setLogCond(0, "", 0, "", 5)
    if (len(sys.argv) != 3):
        print "\nCorrect usage is:\n"
        print "% (python) ngamsBrotherPT9200DxPlugIn <NGAMS CFG> <text>\n"
        sys.exit(1)
    cfg = ngamsConfig.ngamsConfig()
    cfg.load(sys.argv[1])
    ngamsBrotherPT9200DxPlugIn(cfg, sys.argv[2])

#
# ___oOo___
```

Figure 39: Example Label Printer Plug-In (FILE: “ngams/ngamsPlugIns/ngamsBrotherPT9200DxPlugIn.py”).

15 *EXPERT:* The Data Archiving Plug-In - DAPI

The purpose of the DAPI, is to handle the archiving of data files. There are often specific aspects to take into account while archiving various kinds of data. The DAPIs make it possible to adapt NG/AMS for handling new (user specific) types of data. I.e., nothing is hard coded in the SW in connection with the data handling.

When the NG/AMS Server receives an Archive Request, a thread is spawned to handling the request. It first classifies the data and finds the appropriate Storage Set on which to store the file. Subsequently it receives the data into an intermediate file with a unique name in the Staging Area on the Main Disk of the target Storage Set. The target Storage Set is determined from the NG/AMS Configuration. From the mime-type of the data a suitable Stream is found, and afterwards a suitable Storage Set.

After having received the file, the DAPI configured for handling that type of data is invoked to carry out specific tasks to be done during the archiving.

The main tasks of a DAPI are as follows:

- **Data Consistency Checking:** Usually it is advisable to carry out a check of the data before archiving it. Such a check could e.g. be to calculate the checksum of the file, or to check that certain parameters are properly set in the data file. If inconsistencies are found, the file should be moved to the Bad Files Directory on the Target Disk. This however, is done by NG/AMS. If a file is found to be bad, an exception should be thrown, which contains the error mnemonic "NGAMS_ER_BAD_FILE"; see the example DAPI, Section 15.3 for clarification on this topic (FUNCTION: "ngams/ngamsPlugIns/ngamsFitsPlugIn.checkCheckSum()").
- **Data Processing:** Before archiving a file, it is often necessary/required to do some processing. It could be something as simple as compressing the file, but in principle there are no limits to the kind of data processing that can be carried out. If the processing changes the mime-type of the file, it is important that the DAPI returns the new mime-type to NG/AMS.
- **Generating Final (Target) Filename:** The target filename of a data file may be generated from parameters in the header. The filename is composed by the Mount Point of the disk plus the Path Prefix from the configuration. How the rest of the filename is generated is up to the plug-in implementation.
- **Generating Standard DAPI Return Value:** A number of parameters like File ID, File Version, file size, Disk ID and more for the file archived must be returned to NG/AMS in order to update the NGAS DB accordingly. A convenience function provided in "ngamsPlugInApi" should be used for this (FUNCTION: “ngams/ngamsPlugInApi.genDapiSuccessStat()”).

After the DAPI has finished execution, NG/AMS will move the processed file to its final destination (which was decided by the DAPI). Also the NGAS DB is updated by NG/AMS with the information about the new file. If replication is requested, the file is replicated and the DB updated, also with the information for the Replication File.

The DAPI is only concerned with the Main File. If a Replication File should be produced this is entirely handled by NG/AMS. The DAPI can indicate to NG/AMS that no replication should be carried out by setting the flag “ngamsReqProps.setNoReplication(1)”. Note, that if no Replication Disk is specified in the Storage Set, no replication is performed automatically. If replication is switched off via the configuration file, by the DAPI or if no Replication Disk is specified in the configuration, the information about the Replication File is not updated in the DB.

File Versioning can be switched off by the client issuing the file for archiving using the parameter “no_versioning=1” (see 27.1). The DAPI must use the value of “ngamsReqProps.getNoVersioning()” to check if File Versioning is active. This is handled automatically by the convenience function ”ngamsPlugInApi.genFileInfo()” and the DAPI does not need to worry about this.

The diagram in Figure 40 shows the actions carried out by NG/AMS and the DAPI while handling an Archive Request. Only the main actions are shown in the figure. Behind the scenes a number of other tasks are performed in order to archive a file properly.

As seen in Figure 40, the handling of an Archive Request is initiated by a data provider sending an Archive Pull or an Archive Push Request to the NG/AMS Server (1). NG/AMS determines the type of data (mime-type)[7] and from this the Target Storage Set is determined. Subsequently the data is received into the Staging Area on the Target Main Disk (2). Subsequently the DAPI is invoked (3), which does the necessary data consistency checking, processing and extraction of

[7] If the mime-type is not specified explicitly in the Archive Request, NG/AMS will attempt to determine the mime-type from the extension specified in the URI of the data file issued for archiving. In this case the value of the mime-type should be set to the generic mime-type “ngas/archive-request” to signal to NG/AMS to figure out the mime-type.

information from the file (4). The DAPI returns control to NG/AMS and delivers back a set of information needed by NG/AMS for the further processing of the file (5). NG/AMS stores the Main File in its final location on the Main Disk (6). Then the information about the new Main File is updated in the NGAS DB (7). If replication is enabled and a Replication Disk is defined, NG/AMS creates the Replication File (8). Afterwards the information for the Replication File is updated in the DB (9). NG/AMS can either return an immediate reply to the client issuing the Archive Request or it can return a reply when the file has been successfully (or unsuccessfully) handled.

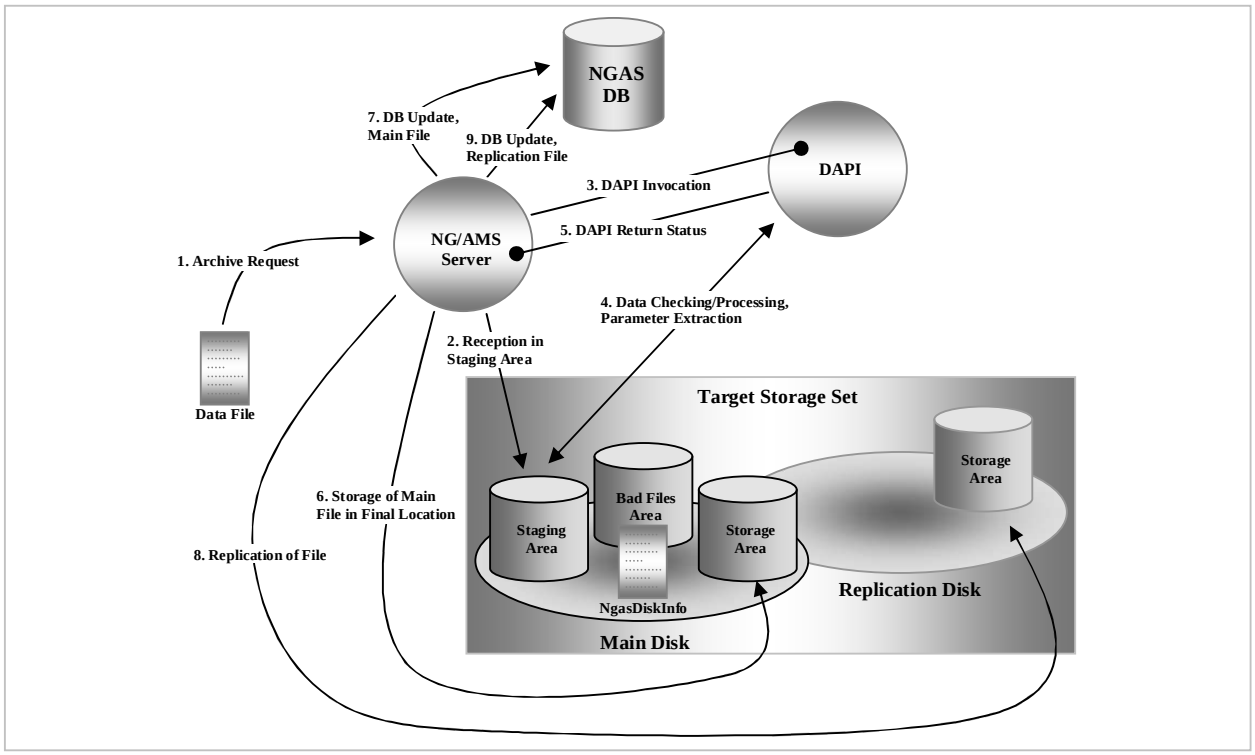

Figure 40: Handling of an Archive Request.

Note, that the DAPI is a function running within the same Python interpreter as the NG/AMS Server process.

15.1 *EXPERT:* Interface of a DAPI

The DAPI must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual DAPI, which is invoked by NG/AMS.

A DAPI has an interface as shown in Figure 41.

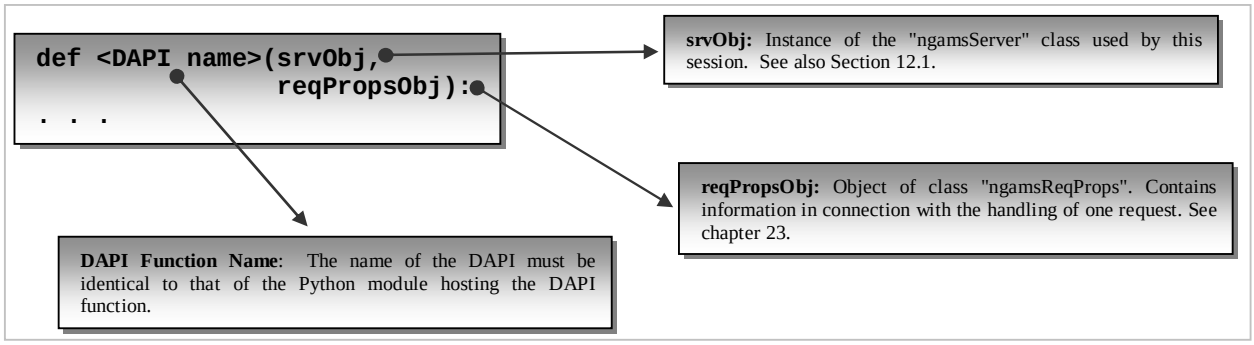

Figure 41: Function interface of a DAPI.

A DAPI must perform the following return when finishing execution:

```
return ngamsPlugInApi.genDhpiSuccessStat(diskId,
                                         relFilename,
                                         fileId,
                                         fileVersion,
                                         format,
                                         fileSize,
                                         uncomprSize,
                                         compression,
                                         relPath,
                                         slotId,
                                         fileExists,
                                         completeFilename)
```

Figure 42: DAPI return statement.

The return parameters of a DAPI are as follows:

Parameter	*Type*	*Description*
`diskId`	String	Disk ID of file.
`relFilename`	String	Filename relative to mount point.
`fileId`	String	File ID allocated to the file by the DAPI.
`fileVersion`	Integer	Version of the file.
`format`	String.	Format (or mime-type) of the file. Only mime-types defined in the NG/AMS Configuration are accepted.
`fileSize`	Integer	Size of the file as it is archived.
`uncomprSize`	Integer	Uncompressed size of the file. I.e., if the file was compressed, this is the original size before archiving/compression.
`compression`	String	Compression method used to compress file. Should be the command invoked to compress the file, e.g. "compress".
`relPath`	String	Path relative to the mount point of the target disk.
`slotId`	String	Slot ID of slot in which the Main Disk is installed.
`fileExists`	Integer	Indicates if the file already existed on the target disk. In case yes, this should be 1, otherwise 0.
`completeFilename`	String	The complete name of the file as it should be. The complete name must be generated by the DAPI.

Table 22: Return parameters of a DAPI.

15.2 *EXPERT:* Overall Structure & Algorithm of a DAPI

The overall structure of a DAPI Python source and in particular the DAPI function itself is shown in Figure 43.

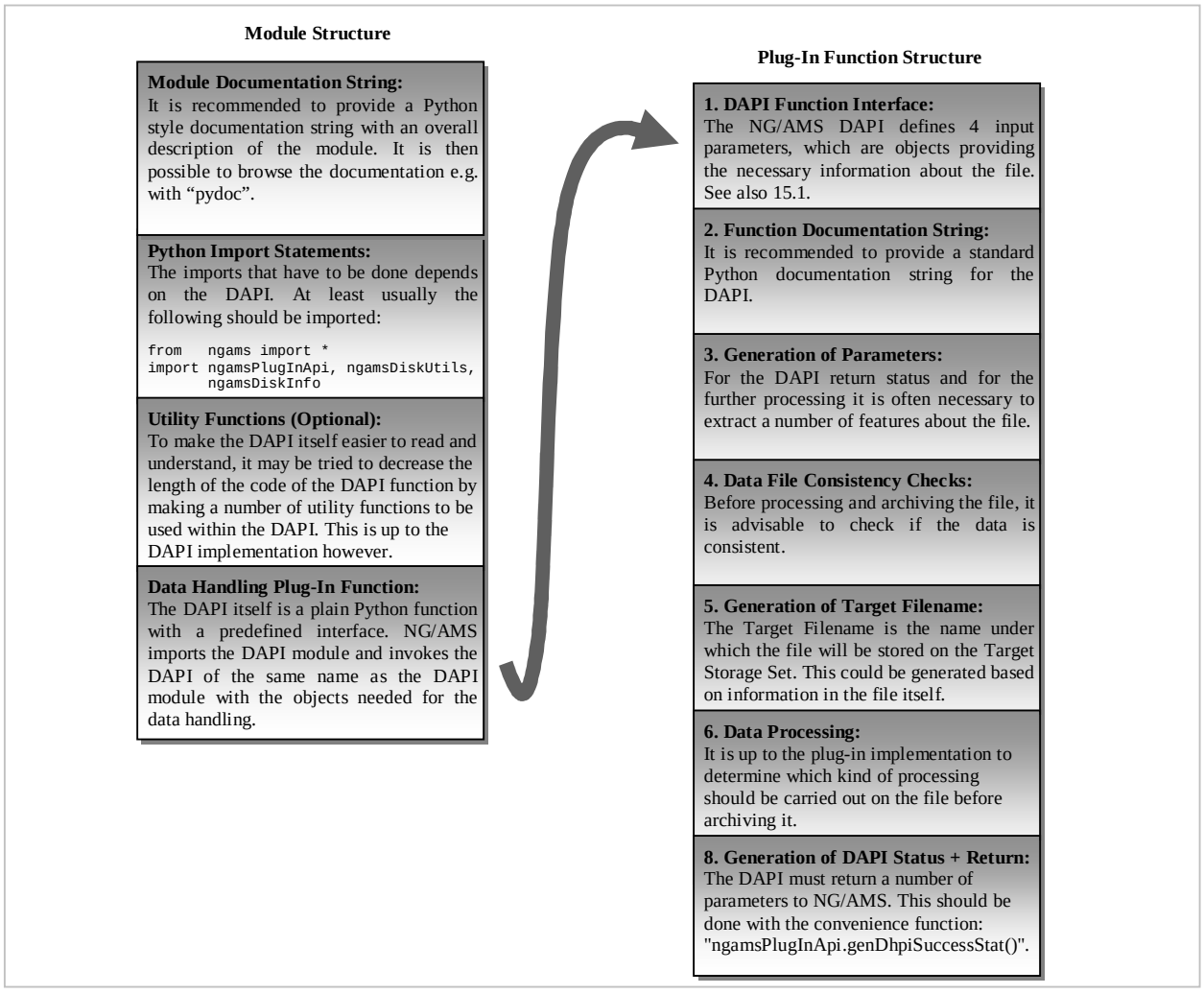

Figure 43: Typical structure of a DAPI module and a DAPI function.

The exact sequence of the actions performed and the actions themselves, may vary from DAPI to DAPI. I.e., maybe the data processing is done before the generation of the final target filename. In Section 15.3 an example DAPI module is shown.

15.3 *EXPERT:* Example DAPI - WFI/FITS File DAPI

In the following an example DAPI, which is used for archiving FITS files at the 2.2m telescope at La Silla is shown:

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsFitsPlugIn.py,v 1.50 2002/07/10 08:34:33 arcsw Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  10/05/2001  Created
#

"""
This Data Archiving Plug-In is used to handle reception and processing
of FITS files.

Note, that the plug-in is implemented for the usage at ESO. If used in other
contexts, a dedicated plug-in matching the individual context should be
implemented and NG/AMS configured to use it.
"""

import os, exceptions, string
import PccUtTime
from   ngams import *
import ngamsPlugInApi, ngamsDiskUtils, ngamsDiskInfo


def getDpIdInfo(filename):
    """
    Generate the File ID (here DP ID) for the file.

    filename:   Name of FITS file (string).

    Returns:    Tuple containing the value of ARCFILE, the DP ID
                of the file, and the JD date. The two latter deducted from
                the ARCFILE keyword (tuple).
    """
    try:
        keyDic  = ngamsPlugInApi.getFitsKeys(filename, ["ARCFILE"])
        arcFile = keyDic["ARCFILE"][0]
        els     = string.split(arcFile, ".")
        dpId    = els[0] + "." + els[1] + "." + els[2]
        date    = string.split(els[1], "T")[0]
        # Make sure that the files are stored according to JD
        # (one night is 12am -> 12am).
        isoTime = els[1]
        ts1 = PccUtTime.TimeStamp(isoTime)
        ts2 = PccUtTime.TimeStamp(ts1.getMjd() - 0.5)
        dateDirName = string.split(ts2.getTimeStamp(), "T")[0]

        return [arcFile, dpId, dateDirName]
    except:
        err = "Did not find keyword ARCFILE in FITS file"
        errMsg = genLog("NGAMS_ER_BAD_FILE", [os.path.basename(filename),
                                              "ngamsFitsPlugIn", err])
        raise exceptions.Exception, errMsg


def checkFitsFileSize(filename):
    """
    Check if the size of the FITS file is a multiple of 2880. If this
    is not the case, we through an exception.

    filename:   FITS file to check (string).

    Returns:    Void.
    """
    if (string.split(filename, ".")[-1] == "fits"):
        size = ngamsPlugInApi.getFileSize(filename)
        testVal = (size / 2880.0)
        if (testVal != int(testVal)):
            errMsg = "The size of the FITS file issued " +\
                     "is not a multiple of 2880! Rejecting file!"
            errMsg = genLog("NGAMS_ER_BAD_FILE", [os.path.basename(filename),
                                                  "ngamsFitsPlugIn", errMsg])
            raise exceptions.Exception, errMsg


def checkChecksum(parDic,
                  filename):
    """
```

```
    Check that the checksum of the file is correct.

    parDic:    Dictionary with disk information (ngamsPhysDiskInfo objects)
               (dictionary).

    filename:  Name of FITS file (string).

    Returns:   Void.
    """
    # Only do check if the checksum_util parameter is set.
    if (not parDic.has_key("checksum_util")): return

    # Execute the checksum routine and evaluate result.
    info(2,"Invoking checksum test utility: " + parDic["checksum_util"] +\
         " on file: " + filename)
    res = ngamsPlugInApi.execCmd(parDic["checksum_util"] + " " + filename)
    if (int(res[0]) != 0):
        errMsg = "Problem occurred invoking checksum check utility: " +\
                 parDic["checksum_util"]
        errMsg = genLog("NGAMS_ER_DAPI", [errMsg])
        error(errMsg)
        raise exceptions.Exception, errMsg
    if (res[1] != parDic["checksum_result"]):
        errMsg = "Executing checksum utility: " + parDic["checksum_util"] +\
                 " gave unexpected result. Result: [" + res[1] + "]. " +\
                 "Expected Result: [" + parDic["checksum_result"] + "]."
        errMsg = genLog("NGAMS_ER_BAD_FILE", [filename, "ngamsFitsPlugIn",
                                              errMsg])
        error(errMsg)
        raise exceptions.Exception, errMsg


# DAPI function.
def ngamsFitsPlugIn(srvObj,
                    reqPropsObj):
    """
    Data Archiving Plug-In to handle archiving of FITS files.

    srvObj:       Reference to NG/AMS Server Object (ngamsServer).

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    Returns:      Standard NG/AMS Data Archiving Plug-In Status
                  as generated by: ngamsPlugInApi.genDapiSuccessStat()
                  (ngamsDapiStatus).
    """
    stagingFilename = ""
    trgFilename = ""
    mountPoint = ""
    info(1,"Plug-In handling data for file with URI: " +
         os.path.basename(reqPropsObj.getFileUri()))
    diskInfo = reqPropsObj.getTargDiskInfo()
    parDic = ngamsPlugInApi.parseDapiPlugInPars(srvObj.getCfg(),
                                                reqPropsObj.getMimeType())
    # If the file is already compressed, we have to decompress it.
    tmpFn = reqPropsObj.getStagingFilename()
    if ((tmpFn.find(".Z") != -1) or (tmpFn.find(".gz") != -1)):
        ngamsPlugInApi.execCmd("gunzip " + tmpFn)
        reqPropsObj.setStagingFilename(os.path.splitext(tmpFn)[0])
    stagingFilename = reqPropsObj.getStagingFilename()
    comprExt = ""
    if (parDic.has_key("compression")):
        if (string.split(parDic["compression"], " ")[0] == "compress"):
            comprExt = "Z"
        elif (string.split(parDic["compression"], " ")[0] == "gzip"):
            comprExt = "gz"

    # Check file (size + checksum).
    checkFitsFileSize(stagingFilename)
    checkChecksum(parDic, stagingFilename)

    # Get various information about the file being handled.
    dpIdInfo = getDpIdInfo(stagingFilename)
    dpId = dpIdInfo[1]
    dateDirName = dpIdInfo[2]
    fileVersion, relPath, relFilename,\
                 complFilename, fileExists =\
                 ngamsPlugInApi.genFileInfo(srvObj.getDb(), srvObj.getCfg(),
                                            reqPropsObj, diskInfo,
                                            stagingFilename, dpId, dpId,
                                            [dateDirName], [comprExt])

    # If a compression application is specified, apply this.
    uncomprSize = ngamsPlugInApi.getFileSize(stagingFilename)
    if (parDic["compression"] != ""):
```

```
        info(2,"Compressing file using: " + parDic["compression"] + " ...")
        exitCode, stdOut, stdErr =\
                  ngamsPlugInApi.execCmd(parDic["compression"] +\
                                         " " + stagingFilename)
        if (exitCode != 0):
            errMsg = "ngamsFitsPlugIn: Problems during data handling! " +\
                     "Compressing the file failed"
            raise exceptions.Exception, errMsg
        stagingFilename = stagingFilename + "." + comprExt
        # Remember to update the Temporary Filename in the Request
        # Properties Object.
        reqPropsObj.setStagingFilename(stagingFilename)
        info(2,"File compressed")

    # Generate status.
    info(4,"Generating status ...")
    format = "application/x-cfits"
    fileSize = ngamsPlugInApi.getFileSize(stagingFilename)
    info(3,"DAPI finished processing of file")
    return ngamsPlugInApi.genDapiSuccessStat(diskInfo.getDiskId(), relFilename,
                                             dpId, fileVersion, format,
                                             fileSize, uncomprSize,
                                             parDic["compression"], relPath,
                                             diskInfo.getSlotId(), fileExists,
                                             complFilename)

#
# ___oOo___
```

Figure 44: Example Data Archiving Plug-In (FILE: “ngams/ngamsPlugIns/ngamsFitsPlugIn.py”).

16 *EXPERT:* The Register Plug-In

The Register Plug-In is used when executing a REGISTER command (see Section 27.8), to handle the processing and extraction of information from a data file, which is being registered. The plug-in is *very* similar to the Data Archiving Plug-In (Chapter 15), but due to a few, minor differences, it has been chosen to define explicitly a new type of plug-in for the purpose of registering files. The main difference between registering and archiving of files, is that when registering files, the files stay in the location where they are, and it is not necessary to create a new target filename for these, and to move these around.

A Register Plug-In must be defined for each type of data that is going to be registered using the REGISTER command. See also Section 6.2/"Register" Element.

16.1 *EXPERT:* Interface of a Register Plug-In

The interface of a Register Plug-In is identical to that of the Data Archiving Plug-In; see Section 15.1 for further information.

16.2 *EXPERT:* Example Register Plug-In

In the following an example Register Plug-IN, which is used for archiving FITS files is shown:

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsFitsRegPlugIn.py,v 1.5 2002/07/10 08:34:33 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  10/05/2001  Created
#

"""
This Data Register Plug-In is used to handle the registration of FITS files
already stored on an 'NGAS disk', which just need to be registered in the DB.

Note, that the plug-in is implemented for the usage at ESO. If used in other
contexts, a dedicated plug-in matching the individual context should be
implemented and NG/AMS configured to use it.
"""

import os, exceptions, string
from   ngams import *
import ngamsPlugInApi, ngamsDiskUtils, ngamsDiskInfo, ngamsFitsPlugIn


# Data Registration Function.
def ngamsFitsRegPlugIn(srvObj,
                       reqPropsObj):
    """
    Data Registration Plug-In to handle registration of FITS files.

    srvObj:       Reference to NG/AMS Server Object (ngamsServer).

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    Returns:      Standard NG/AMS Data Archiving Plug-In Status as generated
                  by: ngamsPlugInApi.genDapiSuccessStat() (ngamsDapiStatus).
    """
    info(1,"Plug-In registering file with URI: " + reqPropsObj.getFileUri())
    diskInfo = reqPropsObj.getTargDiskInfo()
    parDic = ngamsPlugInApi.parseRegPlugInPars(srvObj.getCfg(),
                                               reqPropsObj.getMimeType())
    stageFile = reqPropsObj.getStagingFilename()

    # If the file is already compressed, we have to decompress it.
    if ((stageFile.find(".Z") != -1) or (stageFile.find(".gz") != -1)):
        workingFile, procDir = ngamsPlugInApi.prepProcFile(srvObj.getCfg(),
                                                           stageFile)
        ngamsPlugInApi.execCmd("gunzip " + workingFile)
        if (workingFile.find(".Z") != -1):
            workingFile = workingFile[:-2]
        else:
            workingFile = workingFile[:-3]
    else:
        workingFile = stageFile
```

```
    # Check file (size + checksum).
    ngamsFitsPlugIn.checkFitsFileSize(workingFile)
    ngamsFitsPlugIn.checkChecksum(parDic, workingFile)

    # Get various information about the file being handled.
    arcFile, dpId, dateDirName = ngamsFitsPlugIn.getDpIdInfo(workingFile)
    fileVersion, relPath, relFilename,\
                 complFilename, fileExists =\
                 ngamsPlugInApi.genFileInfoReg(srvObj.getDb(), srvObj.getCfg(),
                                               reqPropsObj, diskInfo,
                                               stageFile, dpId)

    # Generate status.
    info(4,"Generating status ...")
    fileSize = ngamsPlugInApi.getFileSize(stageFile)
    if (stageFile.find(".Z") != -1):
        format = "application/x-cfits"
        compresion = "compress"
    elif (stageFile.find(".gz") != -1):
        format = "application/x-gfits"
        compresion = "gzip"
    else:
        format = "image/x-fits"
        compresion = ""
    uncomprSize = ngamsPlugInApi.getFileSize(workingFile)
    info(3,"Register Plug-In finished processing of file")
    return ngamsPlugInApi.genRegPiSuccessStat(diskInfo.getDiskId(),relFilename,
                                              dpId, fileVersion, format,
                                              fileSize, uncomprSize,compresion,
                                              relPath, diskInfo.getSlotId(),
                                              fileExists, complFilename)

#
# ___oOo___
```

Figure 45: Example Register Plug-In (FILE: “ngams/ngamsPlugIns/ngamsFitsRegPlugIn.py”).

17 *EXPERT:* The Data Processing Plug-In - DPPI

The purpose of the Data Processing Plug-In (DPPI) is to provide a specific type of processing to be applied on a specific type of data when data is being retrieved from an NGAS Host. Processing could be as trivial as simply uncompressing a data file, which is stored in compressed format. It could also be far more complex and involve advanced image processing and parameter extraction. How the DPPI actually processes the data, is left up to the DPPI implementation. The DPPI only has to obey the set of rules for interfacing as for any other plug-in defined for NG/AMS.

17.1 *EXPERT:* Interface of a DPPI

The DPPI must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual DPPI, which is invoked by NG/AMS.

A DPPI has an interface as shown in Figure 46.

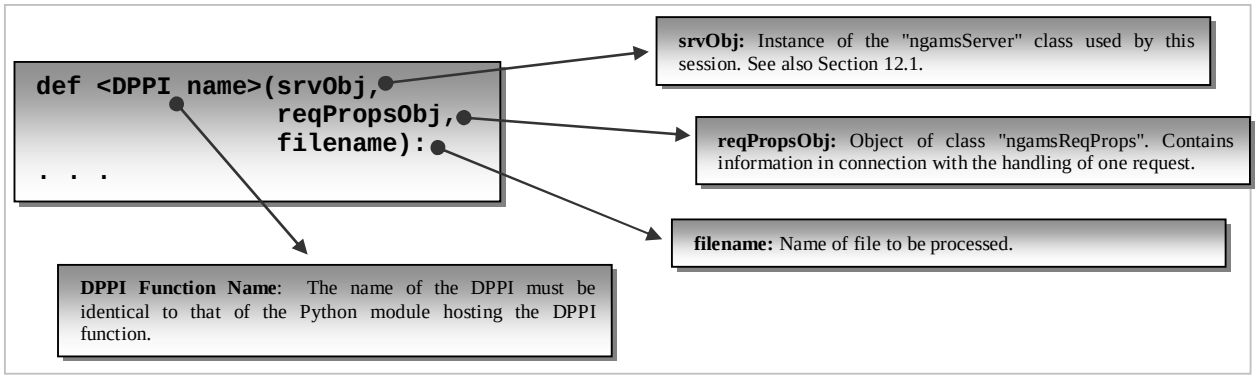

Figure 46: Function interface of a DPPI.

A DPPI must return an object of the type "ngamsDppiStatus". This again contains one or more objects of the type "ngamsDppiResult", which each refer to result data or contains the result of the processing. This means that it is possible to produce several results in a DPPI, and to have these send back to the requestor[8]. The concept of the DPPI return object is shown in Figure 47.

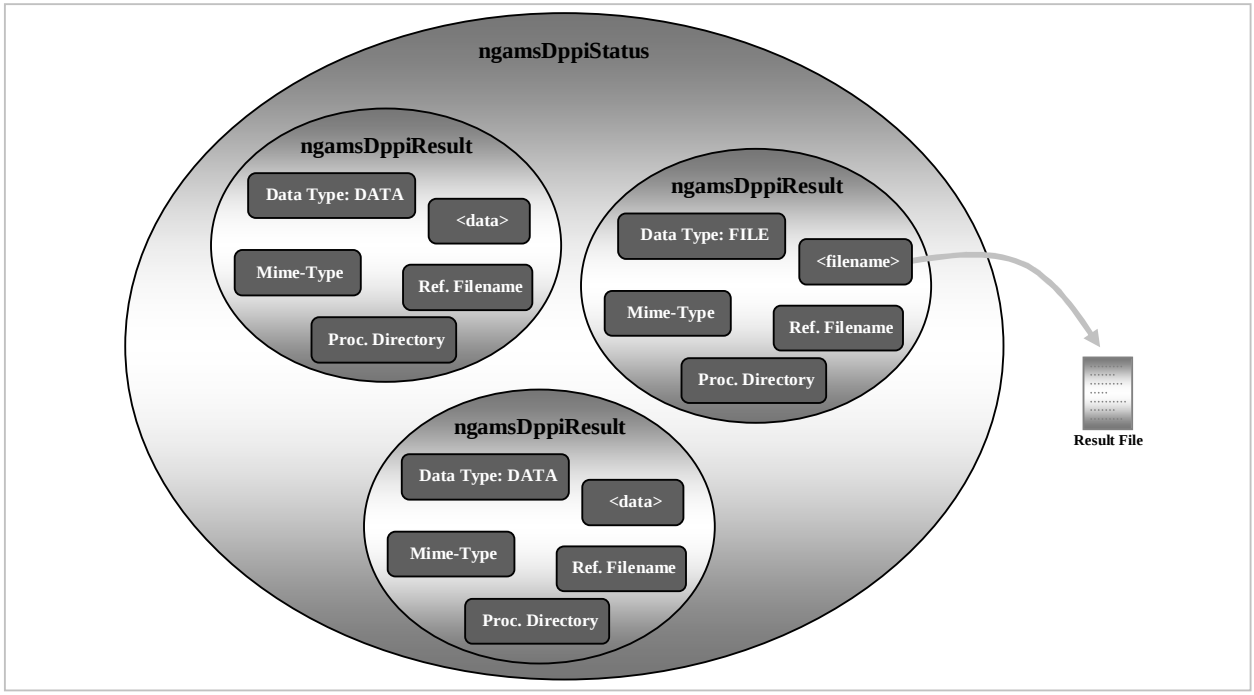

Figure 47: DPPI – structure of return data.

As shown in Figure 47, the "ngamsDppiStatus" object can contain an arbitrary number of "ngamsDppiResult" objects, each containing the information of one sub-result. As can be seen in the figure, the data of a sub result can either be contained directly in the ngamsDppiResult object, or the data can be stored in an external file, which is referred to by the object. Whether to use the one or the other depends on the nature of the data. If the result consists of a smaller amount of non-binary data it is more convenient to store the data internally to avoid having to create, access and delete the result files. For larger amounts of result data and for binary data, it is recommended to use an external result file. See Chapter 23 for more information about these classes.

External, temporary files (Result Files) will be deleted automatically by NG/AMS after the result data has been returned to the requestor.

17.2 *EXPERT:* Example DPPIs

In the following a very trivial example of a DPPI is shown. It is used to extract the header information of a FITS file.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsExtractFitsHdrDppi.py,v 1.9 2002/09/27 12:13:34 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# awicenec  26/09/2002  Created
#

"""
Contains a DDPI which is used to extract the main header from FITS files.
"""

from ngams import *
import ngamsPlugInApi, ngamsDppiStatus
```

[8] For actually supporting completely latter, NG/AMS needs to be extended to be able to return replies making use of the "multipart/mixed" mime-type as known from e.g. emails. This is foreseen to be supported soon.

```
from ngasUtils import printhead

def ngamsExtractFitsHdrDppi(srvObj,
                            reqPropsObj,
                            filename):
    """
    This DPPI extracts the main header from a FITS file requested from
    the ESO Archive.

    Note: This DPPI works directly on the archived file, since it is
          read-only access.

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    filename:      Name of file to process (string).

    Returns:       DPPI return status object (ngamsDppiStatus).
    """
    statusObj = ngamsDppiStatus.ngamsDppiStatus()

    pH = printhead.PrintHead(filename)
    pos = filename.rfind('.fits')
    file_id = filename[:pos]
    resFilename = file_id + '.hdr'
    mimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), resFilename)
    resObj = ngamsDppiStatus.ngamsDppiResult(NGAMS_PROC_DATA, mimeType,
                                             pH.HEAD, resFilename, '')
    statusObj.addResult(resObj)

    return statusObj

#
# ___oOo___
```

Figure 48: Example Data Processing Plug-In (FILE: "ngams/ngamsPlugIns/ngamsExtractFitsHdrDppi.py").

Another example of a trivial DPPI is shown in xxx. This DPPI is used to decompress files, which have been archived in compressed format.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsEsoArchDppi.py,v 1.8 2002/11/04 12:28:47 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  08/01/2002  Created
#

"""
Contains a DDPI which is used by the ESO Archive Facility to perform the
processing in connection with a standard data request handling.
"""

from ngams import *
import ngamsPlugInApi, ngamsDppiStatus

def ngamsEsoArchDppi(srvObj,
                     reqPropsObj,
                     filename):
    """
    This DPPI performs the processing neccessary for the files
    requested from the ESO Archive (by the Data Requestor).

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    filename:      Name of file to process (string).

    Returns:       DPPI return status object (ngamsDppiStatus).
    """
    statusObj = ngamsDppiStatus.ngamsDppiStatus()

    # Decompress the file if the last extension is "Z".
    if (filename.split(".")[-1] == "Z"):
        procFilename, procDir = ngamsPlugInApi.prepProcFile(srvObj.getCfg(),
                                                            filename)
        exitCode, stdOut, stdErr = ngamsPlugInApi.\
```

```
                                execCmd("uncompress " + procFilename)
        if (exitCode != 0):
            errMsg = "ngamsEsoArchDppi: Problems during data handling! " +\
                     "Decompressing the file: " + filename + " failed. " +\
                     "Error message: " + str(stdErr)
            raise exceptions.Exception, errMsg
        resFilename = procFilename[0:-2]
    else:
        resFilename = filename
        procDir = ""
    mimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), resFilename)
    resObj = ngamsDppiStatus.ngamsDppiResult(NGAMS_PROC_FILE, mimeType,
                                             resFilename, resFilename, procDir)
    statusObj.addResult(resObj)

    return statusObj

#
# ___oOo___
```

Figure 49: Example Data Processing Plug-In (FILE: “ngams/ngamsPlugIns/ngamsEsoArchDppi.py”).

18 *EXPERT:* The Data Checksum Plug-In

The Data Checksum Plug-In is a simple plug-in used to generate the checksum value for a data file being archived. This value is written in the record for the file in the NGAS DB, and used later on to check periodically if the file is in a 'good condition'. I.e., that it is not damaged or corrupted in any way. The Data Checksum Plug-In is invoked by NG/AMS after the DAPI has finished the data type specific processing.

18.1 *EXPERT:* Interface of a Data Checksum Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Data Checksum Plug-In has an interface as shown in Figure 50.

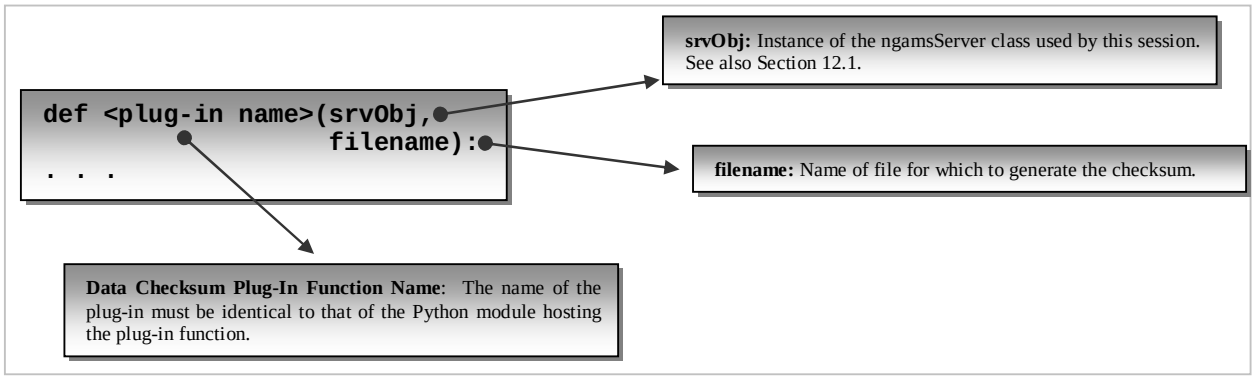

Figure 50: Function interface of a Data Checksum Plug-In (DCPI).

A Data Checksum Plug-In must return the calculated checksum value as a string.

18.2 *EXPERT:* Example Data Checksum Plug-In

In the following the source code of a small example Data Checksum Plug-In is shown. It generates the checksum based on routines built-into Python.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsGenCrc32.py,v 1.12 2002/07/10 08:34:33 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ------------------------------------------------------
# jknudstr  23/01/2002  Created
#

"""
Checksum Plug-In to generate the checksum stored in the ngas_files tables
in connection with each file archived into NGAS.
"""

import binascii
from   ngams import *


def ngamsGenCrc32(srvObj,
                  filename):
    """
    Plug-in to generate CRC-32 checksum for an archived data file.

    srvObj:       Reference to instance of NG/AMS Server class (ngamsServer).

    filename:     Name of file to generate checksum for (string).

    Returns:      CRC-32 checksum for file (string).
```

```
    """
    fo = open(filename, "r")
    buf = fo.read(65536)
    crc = 0
    while (buf != ""):
        crc = binascii.crc32(buf, crc)
        buf = fo.read(65536)
    fo.close()
    return str(crc)

#
# ___oOo___
```

Figure 51: Example Data Checksum Plug-In (FILE: “ngams/ngamsPlugIns/ngamsGenCrc32.py”).

19 *EXPERT:* The Suspension Plug-In

When an NG/AMS Server is suspending itself after the specified suspension time-out has elapsed (CFG: "NgamsCfg.HostSuspension:IdleSuspensionTime"), it invokes the specified Suspension Plug-In (CFG: "NgamsCfg.HostSuspension:SuspensionPlugIn"), which actually carries out the actions needed to suspend the host (see also 4.3). In the simplest case the Suspension Plug-In might simply invoke a "shutdown" command (on UNIX) as root to shut down the host, but in principle there are no limitations to which kind of actions that are performed.

19.1 *EXPERT:* Interface of a Suspension Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Suspension Plug-In has an interface as shown in Figure 52.

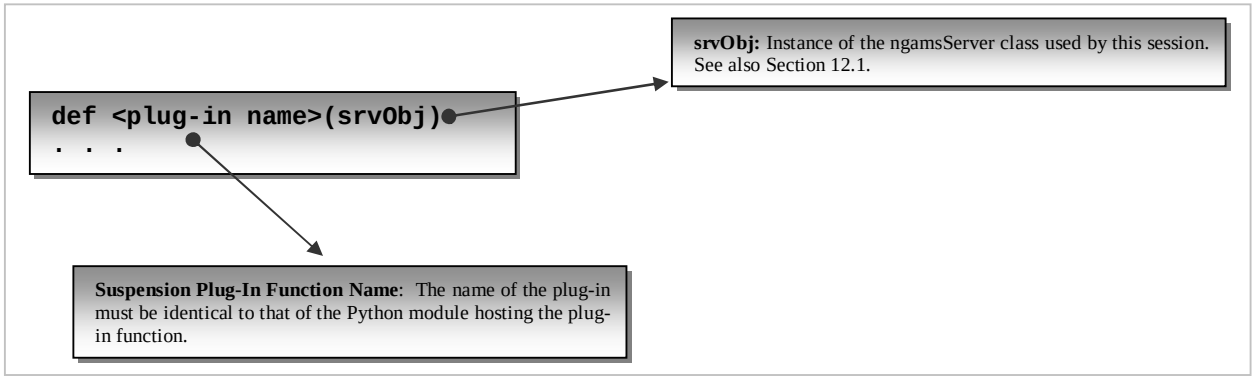

Figure 52: Function interface of a Suspension Plug-In.

A Suspension Plug-In does not return control to the NG/AMS Server but will in fact terminate this after having done various clean-up that might be necessary.

19.2 *EXPERT:* Example Data Checksum Plug-In

```
Under development.
```

Figure 53: Example Suspension Plug-In (FILE: "…").

20 *EXPERT:* The Wake-Up Plug-In

The Wake-Up Plug-In is used by an NG/AMS Server that has been requested to wake-up an NGAS Host that has suspended itself. The Suspension/Wake-Up Service is described in Section 4.3. The actions to be carried out, depends on the HW and on the system configuration. Usually a Wake-Up Plug-In will send a message to a device connected to the network to indicate it to start up an NGAS Host; this device could e.g. be the network card of the host.

20.1 *EXPERT:* Interface of a Wake-Up Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Wake-Up Plug-In has an interface as shown in Figure 54.

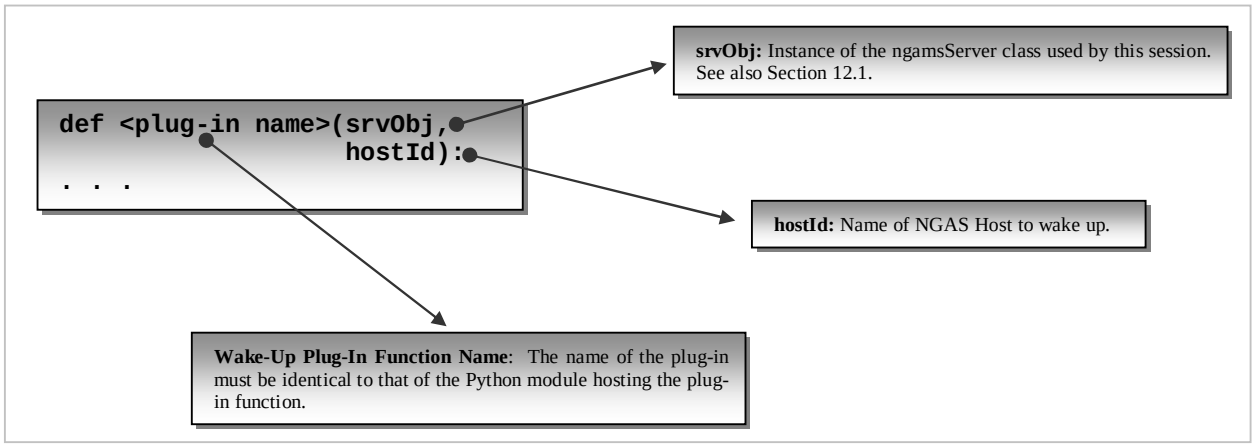

Figure 54: Function interface of a Wake-Up Plug-In.

A Wake-Up Plug-In does not return any value to the NG/AMS Server after execution.

20.2 *EXPERT:* Example Wake-Up Plug-In

```
Under development.
```

Figure 55: Example Wake-Up Plug-In (FILE: "…").

21 *EXPERT:* The Filter Plug-in

The purpose of the Filter Plug-In is to classify data when data is being selected for delivery to a requestor for instance in connection with the Data Subscription Service (see Section 4.2). The Filter Plug-In is a function which understands the data it is applied to, and which based on the contents of the data file, perhaps the filename and other pertinent information can determine if the file matches the requirements.

21.1 *EXPERT:* Interface of a Filter Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Data Checksum Plug-In has an interface as shown in Figure 56.

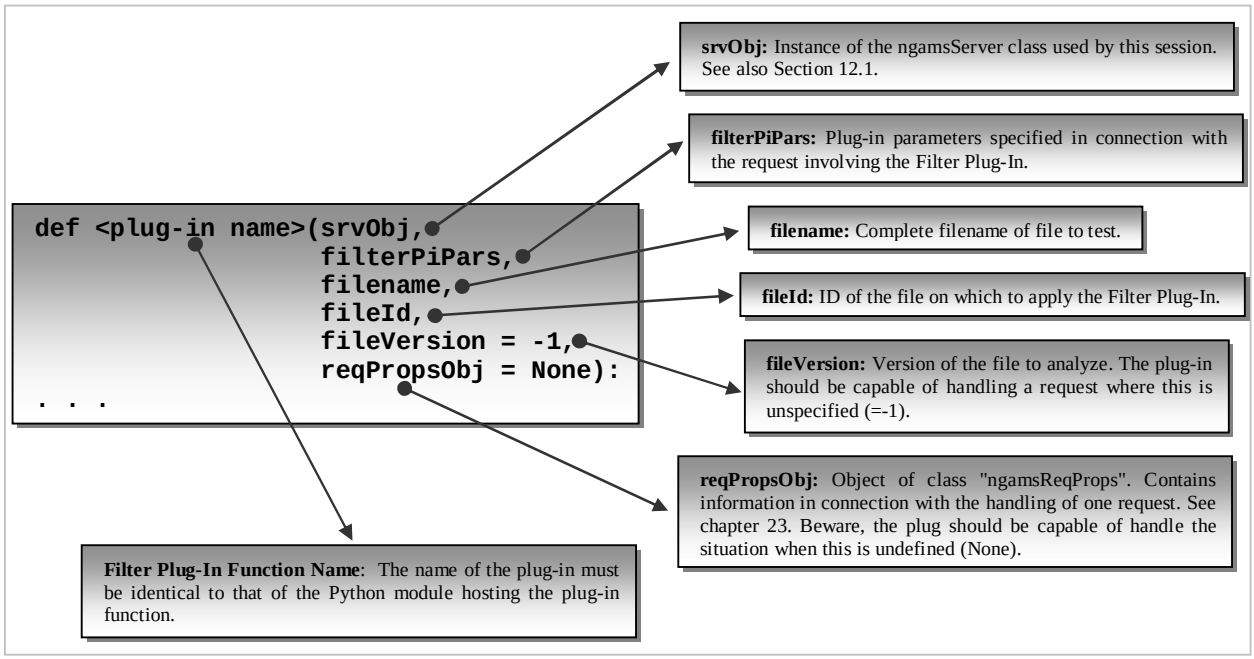

Figure 56: Function interface of a Filter Plug-In.

A Filter Plug-In must return either 1 or 0 depending on whether the filter conditions are met or not.

21.2 *EXPERT:* Example Filter Plug-In

The following Python module, is the implementation of a simple Filter Plug-In, which filters the file specified according to a requested mime-type given in the Plug-In Parameter string ("mime_types", can contain a list of "|" separated mime-types).

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsMimeTypeFilterPI.py,v 1.2 2003/01/02 13:26:10 arcsw Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  21/11/2002  Created
#

import exceptions

from ngams import *
import ngamsPlugInApi


def ngamsMimeTypeFilterPI(srvObj,
                          filterPiPars,
                          filename,
```

```
                        fileId,
                        fileVersion = -1):
    """
    Example Filter Plug-In used to filter on a given mime-type. In case the
    file referenced has the mime-type as specified in the plug-in parameters,
    the file being tested is selected.

    srvObj:        Reference to NG/AMS Server Object (ngamsServer).

    filterPiPars:  Filter Plug-In Parameters (string).

    fileId:        File ID for file to test (string).

    filename:      Filename of (complete) (string).

    fileVersion:   Version of file to test (integer).

    Returns:       0 if the file does not match, 1 if it matches the
                   conditions (integer/0|1).
    """
    match = 0
    parDic = ngamsPlugInApi.parseRawPlugInPars(filterPiPars)
    if (not parDic.has_key("mime_types")):
        errMsg = "ngamsMimeTypeFilterPI: Missing Plug-In Parameter: " +\
                 "mime_types"
        raise exceptions.Exception, errMsg
    refMimeTypes = parDic["mime_types"].split("|")

    # Perform the matching.
    actMimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), filename)
    for mt in refMimeTypes:
        if (actMimeType == mt.strip()): match = 1

    return match

#
# ___oOo___
```

Figure 57: Example Filter Plug-In (FILE: “ngams/ngamsPlugIns/ngamsMimeTypeFilterPI.py”).

22 The NG/AMS Status XML Document

The NG/AMS Status Document is used in various contexts, either as the complete status or as partial status for a specific context. For instance, in the reply of most commands, a small status is given indicating if the command was executed successfully, and in case not, indicating the error that occurred.

22.1 *EXPERT:* NG/AMS Status DTD ("ngamsStatus.dtd")

The NG/AMS Status document is based on the NG/AMS base DTD described in Section 6.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % XmlStd SYSTEM "http://www.eso.org/projects/esoxml/XmlStd.dtd">
%XmlStd;
<!ENTITY % NgamsInternal SYSTEM "ngamsInternal.dtd">
%NgamsInternal;

<!--
  E.S.O.

  Who        When        What
  ********   **********  ************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  ngamsStatus.dtd defines the contents and lay-out of the
  NG/AMS Status Report.

  Consult the DTD ngamsInternal.dtd which contains the actual definition
  of the elements of the NGAMS Status.
  -->


<!--
  The NgamsStatus element is the root element of the NGAMS Status Document.
  -->
<!ELEMENT NgamsStatus (Header, Status, NgamsCfg?, DiskStatus*, FileList*)>


<!--
  The Status Element is used to generate a status with log
  information. It can contain an arbitrary number of log elements
  (defined in the LogMlBase DTD).

  Attributes:
    Date:     Date this Status Element was generated.

    Version:  Version of NG/AMS generating the status.

    HostId:   Name of host where the NG/AMS Server is running.

    Status:   Overall status of the status information. Can be used to
              signal if errors are contained in the Status Elements.

    Message:  Message generated by the NG/AMS Server.

    State:    State of the NG/AMS Server.

    SubState: Sub-State of the NG/AMS Server.
  -->
<!ELEMENT Status  (Trace | Debug | Info | Warning | Error | Alarm | Archive)*>
<!ATTLIST Status  Date             CDATA             #REQUIRED
                  Version          CDATA             #REQUIRED
                  HostId           CDATA             #REQUIRED
                  Status           (OK|FAILURE|-)    "-"
                  Message          CDATA             #IMPLIED
                  State            (ONLINE|OFFLINE)  "OFFLINE"
                  SubState         (IDLE|BUSY)       "IDLE">


<!--
  The NgamsCfg Element contains the configuration used by NG/AMS.
  -->
<!ELEMENT NgamsCfg (Ngams, Server?, Db?, StorageSet*, FileHandling?,
                    Stream*, Monitor?, Log?, Notification?, HostSuspension)>


<!--
  The DiskStatus Element contains the status for each disk and the
  status for the files stored on an HDD.
```

```
  Attributes:
    DiskId:             Unique ID for the HDD.

    Archive:            ID of the archive to which this disk belongs

    InstallationDate:   Date (ISO-8601) the disk was prepared.

    Type:               Type of the HDD.

    LogicalName:        Logical (human readable/memorable) name of
                        the HDD.

    MainDisk:           Indicates if the HDD is the Main Disk or the
                        Replication Disk (0|1).

    HostId:             Name of the host where the HDD is installed.

    SlotId:             Slot ID (number) in which the HDD is installed.

    Mounted:            Indicates if the HDD is mounted in this NGAS
                        System (0|1).

    MountPoint:         Indicates the mount point for the HDD (path).

    NumberOfFiles:      Number of files stored on the disk.

    AvailableMb:        Indicates the capacity of the HDD (MB).

    BytesStored:        Bytes stored on the HDD.

    Completed:          Indicates if the disk is full - no more data
                        should be stored on this disk.

    Checksum:           Checksum for the data on the disk.

    TotalDiskWriteTime: Total time in seconds used for writing the
                        bytes stored on this disk (s).

    LastCheck:          Date for last check.
  -->
<!ELEMENT DiskStatus (FileStatus*)>
<!ATTLIST DiskStatus DiskId              CDATA  #REQUIRED
                     Archive             CDATA  #REQUIRED
                     InstallationDate    CDATA  #REQUIRED
                     Type                CDATA  #REQUIRED
                     LogicalName         CDATA  #REQUIRED
                     MainDisk            (0|1)  #REQUIRED
                     HostId              CDATA  #REQUIRED
                     SlotId              CDATA  #REQUIRED
                     Mounted             (0|1)  #REQUIRED
                     MountPoint          CDATA  #REQUIRED
                     NumberOfFiles       CDATA  #REQUIRED
                     AvailableMb         CDATA  #REQUIRED
                     BytesStored         CDATA  #REQUIRED
                     Completed           (0|1)  #REQUIRED
                     CompletionDate      CDATA  #REQUIRED
                     Checksum            CDATA  #REQUIRED
                     TotalDiskWriteTime  CDATA  #REQUIRED
                     LastCheck           CDATA  #IMPLIED>


<!--
  The FileStatus Element contains the status of each file stored on
  the individual HDD.

  Attributes:
    FileName:               Name of the file (relative to the disk
                            mount point).

    FileId:                 ID of the file, e.g. DP ID for ESO FITS files.

    FileVersion:            File Version.

    Format:                 Mime-type of file.

    FileSize:               Size of the file (bytes).

    UncompressedFileSize:   Size of file uncompressed (bytes). Same as
                            FileSize if uncompressed.

    Compression:            Type of compression applied on the file.

    IngestionDate:          Date the file was ingested (ISO-8601).
  -->
<!ELEMENT FileStatus EMPTY>
```

```
<!ATTLIST FileStatus FileName             CDATA         #IMPLIED
                     FileId               CDATA         #IMPLIED
                     FileVersion          CDATA         #IMPLIED
                     Format               CDATA         #IMPLIED
                     FileSize             CDATA         #IMPLIED
                     UncompressedFileSize CDATA         #IMPLIED
                     Compression          CDATA         #IMPLIED
                     IngestionDate        CDATA         #IMPLIED
                     Ignore               (0|1)         #IMPLIED
                     Checksum             CDATA         #IMPLIED
                     ChecksumPlugIn       CDATA         #IMPLIED
                     FileStatus           CDATA         #IMPLIED>

<!--
  The FileList Element is used to contain a list of files. This can be
  used for various purposes, is e.g. used for the CLONE command to
  indicate which files were attempted to be cloned but which couldn't
  be properly cloned.

  The element contain 1 or more FileStatus Elements.

  Attributes:
    Name:       Name allocated to the list.

    Id:         A short ID allocated to the file list. Should normally
                be a one word string, e.g.: 'PROCESSED_FILES'.

    Comment:    A comment can be added to a File List to remark
                special conditions.

    Status:     A status allocated to the File List.
  -->
<!ELEMENT FileList (FileStatus+ | FileList+)>
<!ATTLIST FileList Id             CDATA         #REQUIRED
                   Comment        CDATA         #IMPLIED
                   Status         CDATA         #IMPLIED>

<!-- oOo -->
```

Figure 58: NG/AMS Status DTD (FILE: “ngams/ngamsData/ngamsStatus.dtd”).

22.2 NGAS Disk Info Status - Example

Apart from keeping the information about an NGAS Disk in the NGAS DB (DB: “ngas_disks”), NG/AMS maintains a snap-host of this information in an XML document on each disk. This document is referred to as the “NgasDiskInfo” file. It is created when the disk is first registered, and subsequently updated each time the NG/AMS Server goes Online/Offline, and when the disk is completed.

The following is an example of an “NgasDiskInfo” file. Such XML status documents are stored on each NGAS disk.

```
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2003-01-02T08:40:23.350" HostId="acngast1" Message="Disk status file"
          Version="v2.0-Beta2/2002-12-04T09:22:53"/>
    <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32300" BytesStored="8709834319" Checksum=""
                Completed="0" CompletionDate="" DiskId="IC35L040AVER07-0-SXPTX093675"
                InstallationDate="2002-11-25T09:48:25.000" LogicalName="FITS-M-000001"
                Manufacturer="IBM" NumberOfFiles="163" TotalDiskWriteTime="905.324898006"
                Type="MAGNETIC DISK/ATA"/>
</NgamsStatus>
```

Figure 59: Example NGAS Disk Info file (FILE: “<mount root point>/<disk mount point>/NgasDiskInfo”).

The NGAS Disk Info files are stored at the following location for each disk: “<mount root point>/<disk mount point>/NgasDiskInfo”.

22.3 NGAS File Info Status - Example

The following is an example of a File Info Status document, which is generated e.g. when archiving a file or when issuing a "STATUS?file_id=<file ID>[&file_version=<file version>]" request:

```
<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
```

```
  <Status Date="2003-01-02T13:48:49.758" HostId="acngast1"
          Message="Successfully handled command STATUS" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
  <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32374" BytesStored="8851908825" Checksum=""
              Completed="0" CompletionDate="" DiskId="IBM-DTLA-305040-YJ0YJ070913"
              HostId="acngast1" InstallationDate="2002-11-25T09:48:25.000" LastCheck=""
              LogicalName="FITS-R-000001" Manufacturer="IBM" MountPoint="/NGAS/data2" Mounted="1"
              NumberOfFiles="164" SlotId="2" TotalDiskWriteTime="350.22437346"
              Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1810827525" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="WFI.2001-09-15T22:49:07.652"
                FileName="saf/2001-09-15/1/WFI.2001-09-15T22:49:07.652.fits.Z"
                FileSize="142074506" FileStatus="00000000" FileVersion="1"
                Format="application/x-cfits" Ignore="0" IngestionDate="2003-01-02T13:48:10.000"
                Tag="" UncompressedFileSize="141546240"/>
  </DiskStatus>
</NgamsStatus>
```

Figure 60: Example File Info Status.

23 *EXPERT:* The NG/AMS Python Modules

In this chapter an overview of the NG/AMS Python modules, classes, functions and 'constants' is given. It is not the intention to provide the complete and detailed documentation for all this. This is contained as inline Python documentation in the Python code and can be browsed online. See Section 23.2 for a description of how to do this. The purpose of this chapter is merely to give an overview of the NG/AMS Package.

For the basic usage of NG/AMS it is normally not necessary to have a deep knowledge about the internals of the SW. However, when developing the different types of plug-ins, which must be provided to adapt NG/AMS to various specific contexts, it is an advantage, and in some cases crucial, to have some insight in and overview of the SW and the classes and features available.

23.1 *EXPERT:* NG/AMS Module Structure

Although this manual is not meant as a maintenance manual for NG/AMS, the structure of the NG/AMS modules is briefly mentioned here. This information may be useful in case of troubleshooting or in general for obtaining a deeper insight into the system. The main NG/AMS project module contains the following files and modules (only items of interest in this context are listed):

Module/File	*Description*
ngams	The main module containing all the NG/AMS source code. This is managed in a CVS repository located within the ESO network.
__init__.py	The main Python module containing definitions of basic functions, and definition of various constants (variables).
LICENSE	File that contains the license and distribution conditions for the NG/AMS SW.
INSTALL	File that provides a small installation guide for the NG/AMS SW.
VERSION	Contains the version information for NG/AMS. This is the information that is printed on stdout when issuing the "-version" parameter to the NG/AMS Server or the command line utilities. This is also used in the XML status messages sent back as response to requests to the server.
ngamsCClient	The NG/AMS C based API. Also provides the C based command line utility.
ngamsData	Contains the definition of the various NG/AMS XML data formats. In addition various example files are provided.
ngamsLib	The base module provides various Python modules with fundamental functions, classes and methods used throughout the NG/AMS SW.
ngamsPClient	The NG/AMS Python based API. Also provides the Python based command line utility.
ngamsPlugIns	Contains various example plug-ins implemented for the usage of NG/AMS within ESO.
ngamsServer	Contains the source code used to build the NG/AMS Server.
ngamsSql	Contains the SQL scripts used to build the NGAS DB.

Table 23: Files and modules in the NG/AMS project.

The "ngamsLib" module is the one, which a plug-in developer mostly will be concerned with, although some knowledge about the NG/AMS Server Class (and Python module) is also needed.

In the following some components of potential interest for plug-in development from the module "ngamsLib" are briefly described:

Python Module	*Class*	*Description*
ngamsConfig.py		Contains the code for the "ngamsConfig" class together with other classes used in connection with the NG/AMS Configuration. This is all used to handle the configuration programmatically.
	ngamsDppiDef	Contains the definition of one DPPI from the configuration.
	ngamsStorageSet	Class used to manage the information in connection with one Storage Set from the NG/AMS Configuration.
	ngamsStream	Class used to manage the information in connection with a Stream Definition from the NG/AMS Configuration.
	ngamsConfig	Class used to handle the information in the NG/AMS Configuration. It is possible to load and save the configuration file, as well as to setting and getting all properties of the configuration. It is also possible to generate an XML document of the configuration contained in the object.
ngamsDb.py		The module provides the class "ngamsDb", which is used to access the NGAS DB. All DB access should be performed through this class. This therefore contains all the necessary SQL queries used by the NG/AMS SW. Many methods are provided to perform various, specific queries into the NGAS DB. A 'native SQL query' can be performed using the method "ngamsDb.query()".
ngamsDapiStatus.py		The module provides the class "ngamsDapiStatus", which is used to handle the status

		information from the execution of a Data Handling Plug-In (see Chapter 15). An instance of this class is returned by a DAPI to the NG/AMS Server.
ngamsDiskInfo.py		Provides the class "ngamsDiskInfo", which is used to handle all the information in connection with an NGAS disk. The object can also contain information about the files on the disk. This is stored internally as "ngamsFileInfo" objects. It is possible to generate an NG/AMS XML Status document from the contents of the object.
ngamsDiskUtils.py		Functions used to carry out the handling/management of the disks installed. Among this are function to extract the information about the disk configuration, and a function to check the accessibility of the disks installed.
ngamsDppiStatus.py		The module provides the class "ngamsDppiStatus", which is used to handle the status information from the execution of a Data Processing Plug-In (see Chapter 17). An instance of this class is returned by a DPPI to the NG/AMS Server.
	ngamsDppiResult	Class that contains a sub-result from a DPPI execution.
	ngamsDppiStatus	Class that contains the resulting data from a DPPI execution.
ngamsFileInfo.py		The module provides the class "ngamsFileInfo", which is used to handle all the information in connection with a file, which has been archived in an NGAS Host. It is possible to generate an XML document from the contents of the object.
ngamsFileList		Used to manage list of file information objects (ngamsFileInfo), e.g. to dump the information into XML documents.
ngamsLib.py		Contains various basic convenience functions used throughout the NG/AMS SW.
ngamsPhysDiskInfo.py		Provides the class "ngamsPhysDiskInfo", which is used to manage the 'physical information' about a disk extracted by the System Online Plug-In (see Chapter 12).
ngamsPlugInApi.py		Modules that provides various utility functions to be used for implementing plug-ins. It is recommended only to use the functions contained in this module for implementing the plug-in, apart from varios classes like ngamServer, ngamsDb and ngamsConfig.
ngamsReqProps.py		Module that provides the object "ngamsReqProps", which is used to keep a record of actions carried out during the handling of a request.
ngamsStatus.py		Provides the class "ngamsStatus", which is used to handle the information in connection with a status generated for NG/AMS.
ngamsSubscriber.py		Used to handle information about one Subscriber.
ngamsUrlLib.py		Modules that provides a small class "ngamsURLopener", which is used to access URLs in a transparent manner.

Table 24: Python modules in the “ngamsLib” sub-module.

23.2 *EXPERT:* Online Browsing of NG/AMS Inline Python Documentation

It is possible to browse online the Python documentation contained in the NG/AMS Python source code files. This provides an acurate and comprehensive description of all classes, methods and functions. The following notation has been used to document the interfaces of methods and functions:

```
def notify(ngamsCfgObj,
           type,
           subject,
           msg):
    """
    Send a notification e-mail to a subscriber about an event happening.

    ngamsCfgObj:   Reference to object containing NG/AMS Configuration file (ngamsConfig).

    type:          Type of Notification (See NGAMS_NOTIF_* in ngams).

    subject:       Subject of message (string).

    msg:           Message to send (string).

    Returns:       Void.
    """
    <code>
```

Figure 61: Example of NG/AMS inline documentation.

First in the description of a method/function, a small description of the task performed by the method is provided. After that the input parameters are listed. After the description of each parameter the type of the parameter is indicated in paranthesis. The return value is also given in connection with the "Returns:" tag.

The documentation can be browsed in an easy manner by using the documentation generator provided together with the Python package ("pydoc"). This can also be used as an HTTP server, e.g.:

```
arcdev1 jknudstr:~/dev/ngams 65 > pydoc -p 7878 &
[2] 15578
arcdev1 jknudstr:~/dev/ngams 66 > pydoc server ready at http://localhost:7878/
```

Figure 62: Starting the pydoc utility as an HTTP server.

Afterwards the NG/AMS documentation can be accessed online via the URL (e.g.):

http://arcdev1.hq.eso.org:7878/ngams.html

The pydoc utility provides a convenient way of browsing the documentation, and generates the documentation online. It locates the NG/AMS module if installed within the search paths compiled into the Python interpreter. Alternatively it should be located in a path defined in the "PYTHON_PATH" environment variable.

24 *EXPERT:* Installation

The installation of an NG/AMS Server *can* be a relatively straightforward and simple procedure if it is not necessary to create an adapted installation for a specific context. Under normal circumstances, the most complex action in this connection might be to configure the server properly. The steps to carry out to obtain a running NG/AMS installation are as follows:

Step/Action	*Description*
Verify Sybase Installation[9]	Verify that a Sybase server is available (Sybase ASE 12). Also check that it is possible to connect to the server from the NGAS Host (with isql). If this is not possible an entry should be added in the "$SYBASE/interfaces" file. It should also be verified that the libraries "libct.a/libct.sl \| .so" are available in "$SYBASE/lib".
Install Python, Check Existing Python Installation	The present version of Python required for NG/AMS is 2.1 (2.1.1). Check that the proper version is installed. If the wrong version is installed, or if Python is not installed at all, it should be downloaded from **http://www.python.org** and installed according to the instructions. Check in particular that the Sybase Python module is available (if Sybase is used).
Get the NG/AMS Python SW	Get the sources of the NG/AMS SW. This can be requested by contacting: **ngast@eso.org**.
Install NG/AMS SW + Configure the Environment	Install the sources simply by copying the NG/AMS root module directory "ngams" to a path contained within the "PYTHON_PATH" list of paths, or add the new location of "ngams" in the "PYTHON_PATH" variable. The NG/AMS C-API should also be compiled and installed. This is done by entering in the directory "ngams/ngamsCClient" and typing "make clean all". The binary "ngamsCClient" should be installed in a 'bin' directory for global access. The "ngams.h" and "libngams.a" files should be copied into an area which is globally accessible (if needed for application development). It could also be chosen to make the NG/AMS Server source file ("ngams/ngamsServer/ngamsServer.py") executable and globally accessible. The same goes for the NG/AMS Python API ("ngams/ngamsPClient/ngamsPClient.py").
Prepare Sybase DB	Prepare the NGAS DB in the Sybase DB server. A user to be used by the NG/AMS when connecting to the DB should be created, e.g.: "ngas". The NGAS tables must also be created. This should be done using the SQL script contained in "ngams/ngamsSql". The script is called: "ngamsCreateTables.sql. The script can be executed using "isql".
Prepare NG/AMS Configuration	Use possibly as a template configuration the configuration example file provided within the NG/AMS SW package ("ngams/ngamsData/ngamsServer.xml"). Go carefully through the list of parameters and configure these according to the description provided in Chapter 6).
Prepare Plug-Ins	Prepare the necessary plug-ins needed for operating NG/AMS. The plug-ins to consider first are the System Online and Offline Plug-Ins; see the Chapters 12 and 13. In addition the DAPI for each type of data to be handled (archived); see Chapter 15. If it is desirable to calculate a checksum for the data files being archived, a Data Checksum Plug-In must be provided; see Chapter 18. If data should be processed, a DPPI should be provided for each type of processing offered by the system; see Chapter 17. If labels for the disk cases should be generated, a Label Printer Plug-In must be provided as well; see Chapter 14. Example implementations of all of these types of plug-in are provided within the NG/AMS package ("ngams/ngamsPlugIns"). Note that all plug-ins provided should be made available in a path pointed to by the "PYTHON_PATH" variable or in one of the search paths compiled into the Python interpreter.
Launch Server in Simulation Mode	The first time when the NG/AMS Server is started after doing all the necessary configuring, it may be convenient to start it manually in an xterm in the Verbose Mode (Verbose Level 3 or 4); see also Section 5.1. This could be done in Simulation Mode to first get the basic things

[9] In the present release only Sybase is supported. Other DBMS' will be supported in the future by means of ODBC.

	straightened out. If the server encounters problems, it will bail out and report these on "stdout". The switching on/off of the Simulation Mode/Normal Mode must be done in the NG/AMS Configuration. It could be tried to issue some commands like ARCHIVE and RETRIEVE to verify the proper functioning.
Launch server in real mode	When the server is running properly in Simulation Mode, it could be tried to switch to the Normal Mode (in the configuration), and try the same as described in the previous step.
Decide how to Start the Server	When the server can be executed and is operating correctly, it should be decided how it should be started. Under normal circumstances it should be started when the host on which it is running is booting, and run as a daemon. I.e., the start-up scripts on the host should be configured accordingly.
Handling of Local Log Files	If a Local Log File is generated, it should be considered that this will continuously grow in size. The speed with which it will be growing, depends on the Log Level selected. If it is desirable to keep the log files, a DAPI to handle this could be provided for NG/AMS and a cron job launched periodically to archive the log file into NG/AMS and subsequently to delete it. If it is not desirable to preserve the information, the file could be deleted periodically. This however, is up to the people responsible for the individual installation to decide how to handle this.
Configuring of Security Mechanisms	Since no security mechanisms are provided at the level of NG/AMS to prevent 'intruders' to connect to the server, such mechanisms should be put in place at the level of the operating system or network. It is up to the people responsible for the security in connection with IT services to decide how to implement this.
Setting Up of Multi-Site DB Environment	If an organization is running a distributed NGAS system, whereby data e.g. are produced on several remote sites, and are made available in an Archive Facility, considerations should be done as how to set up the DB infrastructure. It might be most logical to have the central/reference DB in connection with the Archive Facility, and to set up replication from the various remote sites to the Archive Facility DB. See also Section 8.7 for more information about this issue.

Table 25: Steps needed to install NG/AMS.

As can bee seen above, the installation in the worst-case may be a quite complex procedure. It is therefore not feasible to provide a complete and detailed information in the NG/AMS User's Manual about this. In case of problems or questions it is suggested to contact: **ngast@eso.org** for advice and help on how to approach this matter.

25 NG/AMS Log and Error Messages Definition

Many important log messages (information) and error messages are defined in a formal way in the XML document "ngams/ngamsData/ngamsLogDef.xml". Defining these log messages in this way makes it possible for a client application to better parse/analyze a reply sent back from NG/AMS, since the log definition is based on tags and error code, which will appear in the message. The complete log definition is contained in Table 26.

Context	NGAMS
Release	1.0
Source	jknudstr@eso.org
Revision	@(#) $Id: ngamsLogDef.xml,v 1.10 2002/12/19 09:14:35 arcsw Exp $
Description	This XML document contains the definition of the error logs used within the NG/AMS project.

Log ID	NGAMS_ER_NO_STORAGE_SET
Log Number	1000
Log Type	ERROR
Log Text	No Storage Set matching the Slot ID: %s. Check NG/AMS Configuration: %s.
Description	NG/AMS could not find a Storage Set which matches the given Slot ID. There seems to be a problem in the NG/AMS Configuration.

Log ID	NGAMS_ER_NO_MIME_TYPES
Log Number	1001
Log Type	ERROR
Log Text	No mime-type/extension mappings defined in configuration file: %s (Element: MimeTypes)!
Description	There are no mime-types/Data Handling Plug-In mappings defined in the NG/AMS Configuration. These are necessary in order to have each type of data file properly handled by NG/AMS.

Log ID	NGAMS_ER_MISSING_ELEMENT
Log Number	1002
Log Type	ERROR
Log Text	Could not find element: %s in NG/AMS Configuration: %s. Must be specified!
Description	The element referred to in the log text, was not found in the configuration file as expected. Must be specified in order to run the NG/AMS Server.

Log ID	NGAMS_ER_CRE_GLOB_BAD_FILE_DIR
Log Number	1004
Log Type	ERROR
Log Text	Problem creating Global Bad Files Directory: %s specified in configuration file: %s. Parameter: FileHandling.GlobalBadDirLoc.
Description	The Global Bad Files Directory could not be created. Make the parent directory writable for the NG/AMS Server host account and try again.

Log ID	NGAMS_ER_CONF_PROP

Log Number	1005
Log Type	ERROR
Log Text	%s
Description	The value defined for the property referred is not properly defined. Must define a proper value. Check configuration file and try again.

Log ID	NGAMS_ER_CONF_FILE
Log Number	1006
Log Type	ERROR
Log Text	%s
Description	An error occurred while parsing the configuration file at the position in the document as indicated by the error message. Check/correct configuration file and try again.

Log ID	NGAMS_ER_ILL_MOUNT_ROOT_DIR
Log Number	1007
Log Type	ERROR
Log Text	Illegal path specified for Mount Root Directory in configuration file: %s (Parameter: FileHandling.MountRootDirectory). Path given: %s.
Description	The directory specified as Mount Root Directory is not writable or not existing. Create this and try again.

Log ID	NGAMS_ER_LOAD_CFG
Log Number	1008
Log Type	ERROR
Log Text	Problem encountered attempting to load the NG/AMS Configuration: %s. Error message: %s
Description	The specified configuration file could not be loaded. This could e.g. be due to that the file is not readable (has not read permissions) for the user running NG/AMS, or that the file is not available.

Log ID	NGAMS_ER_ILL_PROC_DIR
Log Number	1009
Log Type	ERROR
Log Text	Illegal path specified for Processing Directory in configuration file: %s (Parameter: FileHandling.ProcessingDirectory). Path given: %s.
Description	The directory specified as Processing Directory is not writable or not existing. Create this and try again.

Log ID	NGAMS_ER_PLUGIN_PAR
Log Number	1010
Log Type	ERROR
Log Text	Plug-In Parameters are improperly formatted: %s. Correct format is: (par)=(value),(par)=(value)...
Description	The Plug-IN Parameters defined in connection with a plug-in are not formatted as expected.

Log ID	NGAMS_ER_MISSING_DISK_ID
Log Number	2000

Log Type	ERROR
Log Text	Error - Disk ID: %s, not found in DB!
Description	Could not find the Disk ID referred, in the DB.

Log ID	NGAMS_WA_DB_CON
Log Number	2001
Log Type	WARNING
Log Text	DB Connection not open - trying to reconnect ...
Description	A problem occurred interacting with the DB. It will be attempted to connect again, to see if problem may be rectified.

Log ID	NGAMS_ER_DB_COM
Log Number	2002
Log Type	ERROR
Log Text	Problems communicating with the DB: %s
Description	A problem occurred interacting with the DB. It may be temporarily impossible to communicate with the DB server. When this situation occurs, the system can buffer frames (if configured to do this). It will be attempted to archive these buffered frames at a later stage.

Log ID	NGAMS_NOTICE_NGAS_HOSTS
Log Number	2003
Log Type	NOTICE
Log Text	Table ngas_hosts in the NGAS DB could not be accessed.
Description	A problem occurred while trying to access the ngas_hosts table in the NGAS DB. This in some cases is accepted as it possible to operate NG/AMS without the availability of ngas_hosts.

Log ID	NGAMS_ER_MAIN_DISK_WRONGLY_USED
Log Number	3000
Log Type	ERROR
Log Text	Disk in slot: %s, with Logical Name: %s, is previously registered as a Main Disk but is now installed in a Replication Disk slot.
Description	When a disk has been registered by the system as being a Main Disk, it should not be attempted to use it later as a Replication Disk. The way to recover from this problem is to install the disk in a Main Disk Slot, together with the Replication Disk with which it was originally registered.

Log ID	NGAMS_ER_REP_DISK_WRONGLY_USED
Log Number	3001
Log Type	ERROR
Log Text	Disk in slot: %s, with Logical Name: %s, has previously been registered as a Replication Disk but is now installed in a Main Disk slot.
Description	When a disk has been registered by the system as being a Replication Disk, it should not be attempted to use it later as a Main Disk. The way to recover from this problem is to install the disk in a Replication Disk Slot, together with the Main Disk with which it was originally registered.

Log ID	NGAMS_ER_DISK_INACCESSIBLE
Log Number	3004
Log Type	ERROR
Log Text	Disk with ID: %s is not accessible (writable).
Description	NG/AMS probes for each disk installed and configured for an archiving system if it is possible to write on the disk. If not an error is returned.

Log ID	NGAMS_ER_NO_TARG_DISKS
Log Number	3005
Log Type	ERROR
Log Text	No target disks found for the Stream(s) with mime-type(s): %s.
Description	For each Data Stream Defined in the configuration file, a target disk must be available in order for the system to go Online (for an archiving unit). If such is not found, the system will not go Online.

Log ID	NGAMS_ER_DISK_STATUS
Log Number	3006
Log Type	ERROR
Log Text	Error querying information for disk with ID: %s - cannot generate disk status on disk!
Description	An error occurred while trying to query information about a disk from the NGAS DB. This means that it is not possible to generate the NGAS Disk Info Status File on the disk. This file is normally generated when the system goes Online/Offline. The status file for the disk in question, may not be up to date. Note, maybe the problem is caused by incorrect information in connection with the disk. Could also be caused by a general problem with the communication with the DB server.

Log ID	NGAMS_ER_OFFLINE_PLUGIN
Log Number	3007
Log Type	ERROR
Log Text	%s
Description	A problem occurred while executing the Offline Plug-In. The system could not be brought properly to Offline State, or some actions may have been skipped.

Log ID	NGAMS_ER_ONLINE_PLUGIN
Log Number	3008
Log Type	ERROR
Log Text	%s
Description	A problem occurred while executing the Online Plug-In. The system could not be brought properly to Offline State, or some actions may have been skipped.

Log ID	NGAMS_ER_INIT_SERVER
Log Number	3009
Log Type	ERROR
Log Text	Problems occurred initializing NG/AMS Server. Error message: %s
Description	An error occurred while initializing the NG/AMS Server. The server could not be prepared for execution, and was terminated. Consult the NG/AMS Logs for further information.

Log ID	NGAMS_ER_MULT_INST
Log Number	3010
Log Type	ERROR
Log Text	Apparently an instance of the NG/AMS Server is running or the server was not shut down properly previously! If it is desirable to force the server to start, use the -force command line parameter.
Description	Another instance of the NG/AMS Server may be running already as it was intended to start a session. Having several NG/AMS Servers running in parallel, may cause conflicts. If it is the intention to star a new session the previous server should be terminated. Otherwise, if it is the intention to run two servers, another communication port number could be specified. A new session can be forced started using the -force command line option.

Log ID	NGAMS_ER_INIT_LOG
Log Number	3011
Log Type	ERROR
Log Text	Problem setting up logging properties! Check if log file: %s can be created! Exception: %s
Description	A problem occurred setting up the properties for the logging. Check the configuration file and/or the command line input parameters and try again.

Log ID	NGAMS_ER_START_HTTP_SERV
Log Number	3012
Log Type	ERROR
Log Text	Problems starting HTTP serving: %s
Description	The HTTP server could not be started. This could because by conflicting port numbers, or other problems in connection with the network set-up.

Log ID	NGAMS_AL_NO_DISKS_AVAIL
Log Number	3013
Log Type	ALERT
Log Text	ALERT -- NO DISKS AVAILABLE IN THIS NGAS SYSTEM! NGAS ID: %s. Host ID: %s. Check HW and try again!
Description	NG/AMS didn't find any disks installed in this system. Normally disks should be available for proper operation of the system. Check the disk configuration and try again.

Log ID	NGAMS_AL_NO_STO_SETS
Log Number	3014
Log Type	ALERT
Log Text	No Storage Sets found for mime-type: %s
Description	No available Storage Set (disk set) was found for storing the data with the Mime-type as referred above in the error message. Check the disk configuration and the NG/AMS Configuration.

Log ID	NGAMS_AL_NO_TARGET_DISK
Log Number	3015
Log Type	ALERT
Log Text	NO SUITABLE TARGET DISK FOUND FOR DATA WITH MIME-TYPE: %s. - PROBABLY

	NEED TO CHANGE DISKS!
Description	No suitable Storage Set (disks) was found for storing the data with the Mime-type as referred above in the error message. Check the disk configuration and the NG/AMS Configuration.

Log ID	NGAMS_INFO_STARTING_SRV
Log Number	3016
Log Type	INFO
Log Text	Starting/initializing NG/AMS Server - Version: %s - Host: %s - Port %d
Description	The NG/AMS Server is initializing and preparing for execution.

Log ID	NGAMS_INFO_TERM_SRV
Log Number	3017
Log Type	INFO
Log Text	NG/AMS Server terminating - Version: %s - Host: %s - Port %d
Description	The NG/AMS Server is cleaning and preparing to terminate execution.

Log ID	NGAMS_ER_EMAIL_NOTIF
Log Number	3019
Log Type	ERROR
Log Text	Problem sending email notification message to: %s sender: %s, using SMTP host: %s. Error: %s.
Description	The system could not send an Email Notification Message to the destination specified using the SMTP host specified. Check if the parameters are correct, in particular if the SMTP host is correctly specified and accessible.

Log ID	NGAMS_INFO_DATA_CHK_STAT
Log Number	3020
Log Type	INFO
Log Text	Number of files checked: %d. Amount of data checked: %.3f MB. Time for checking: %.3fs.
Description	After one Data Checking cycle has been executed, it is possible to configure NG/AMS to generate a log entry in the log outputs (CFG: NgamsCfg.FileHandling:DataCheckLogSummary).

Log ID	NGAMS_ER_ARCH_BACK_LOG_BUF
Log Number	3021
Log Type	ERROR
Log Text	An error occurred while archiving data found in NG/AMS Back-Log Buffer. Error: %s.
Description	An error was encountered when the NG/AMS Janitor Thread tried to archive a file found in the NG/AMS Back-Log Buffer. The symptom/type of the error is indicated in the error message.

Log ID	NGAMS_ER_NO_DISK_SPACE
Log Number	3022
Log Type	ERROR
Log Text	Not enough disk space for creating file: %s, of size: %d bytes.
Description	NG/AMS could not create a file with the given name of the given size.

Log ID	NGAMS_ER_RETRIEVE_KEYS
Log Number	4000
Log Type	ERROR
Log Text	Problem occurred retrieving key(s): %s, from data file: %s.
Description	The keyword cards listed above, could not be found in the file referred. This means that the handling of the file could not be carried out as expected.

Log ID	NGAMS_ER_DAPI
Log Number	4001
Log Type	ERROR
Log Text	%s
Description	An error occurred in the Data Handling Plug-In - the file has not been handled successfully, and has not been archived.

Log ID	NGAMS_ER_REQ_HANDLING
Log Number	4002
Log Type	ERROR
Log Text	Error occurred handling request! Error/exception: %s
Description	A problem occurred while handling a request issued to NG/AMS. The request could not be carried out successfully.

Log ID	NGAMS_ER_ARCHIVE_PUSH_REQ
Log Number	4003
Log Type	ERROR
Log Text	Problem occurred handling Archive Push Request! URI: %s.
Description	A problem occurred while handling an Archive Push Request. The archive request has not been carried out successfully. It should be investigated what caused the problem, and the file should possibly be archived again.

Log ID	NGAMS_ER_ARCHIVE_PULL_REQ
Log Number	4004
Log Type	ERROR
Log Text	Problems occurred handling Archive Pull Request! URI: %s.
Description	A problem occurred while handling an Archive Pull Request. The archive request has not been carried out successfully. It should be investigated what caused the problem, and the file should possibly be archived again.

Log ID	NGAMS_ER_MISSING_URI
Log Number	4005
Log Type	ERROR
Log Text	Must specify a URI for the data file to archive!
Description	When issuing an archive request, a URI must always be specified. This is used as reference to the file, and in the case of an Archive Pull Request, the URI is used to actually pick up the file.

Log ID	NGAMS_ER_UNKNOWN_MIME_TYPE1
Log Number	4006
Log Type	ERROR
Log Text	Could not determine mime-type for data file with URI: %s. Check NG/AMS Configuration.
Description	From the extension of the file URI given, the mime-type could not be determined, i.e., is not among the mime-types defined in the configuration file. Check the configuration file to see if support for this type of data file should be added.

Log ID	NGAMS_ER_IMPROPER_STATE
Log Number	4007
Log Type	ERROR
Log Text	%s not allowed when in State/Sub-State: %s/%s. Allowed State(s)/Sub-State(s) are: %s/%s.
Description	The request issued cannot be handled by NG/AMS when in the State/Sub-State as indicated in the error message. Bring the system to one of the allowed States/Sub-States listed in the error message and try again.

Log ID	NGAMS_ER_ILL_CMD
Log Number	4009
Log Type	ERROR
Log Text	Illegal command: %s received. Rejecting request!
Description	The command issued is not known/accepted by the NG/AMS Server. Check context and try again.

Log ID	NGAMS_ER_ILL_REQ
Log Number	4010
Log Type	ERROR
Log Text	This NG/AMS is not configured for accepting %s Requests. Rejecting request!
Description	This installation of NG/AMS, is not configured for accepting the request type as specified in the error message. Re-consider the request issued, or to configure the server to handle the given type of request.

Log ID	NGAMS_ER_RETRIEVE_CMD
Log Number	4011
Log Type	ERROR
Log Text	Incorrect parameter given for RETRIEVE command.
Description	The syntax specified for the RETRIEVE command is illegal.

Log ID	NGAMS_ER_UNKNOWN_FILE
Log Number	4012
Log Type	ERROR
Log Text	File with ID: %s is not available. Rejecting request!
Description	It was requested by a client to retrieve information about a certain file, or to return the file. This file however, is not available. Check the File ID and try again.

Log ID	NGAMS_ER_UNKNOWN_DISK

Log Number	4013
Log Type	ERROR
Log Text	Could not retrieve information about disk with Disk ID: %s. Disk is probably unknown to this NG/AMS! Rejecting request!
Description	NG/AMS was requested by a client to return information about a specific disk. This disk however, is not known to NG/AMS. Check the Disk ID (or other information given with the query) and try again.

Log ID	NGAMS_ER_ILL_RETRIEVE_REQ
Log Number	4014
Log Type	ERROR
Log Text	Illegal Disk ID found: %s for file with ID: %s.
Description	The Disk ID registered in connection with the file with the given ID, is not know to NG/AMS. Check DB for inconsistent entries and try again.

Log ID	NGAMS_WA_BUF_DATA
Log Number	4015
Log Type	WARNING
Log Text	Problems occurred while handling file with URI: %s. Data will be buffered, and attempted archived at a later stage. Previous error stack: %s.
Description	A problem occurred while handling an Archive Request. The kind of error that occurred, may be recovered later. The data has therefore been buffered by NG/AMS, and will be attempted archived (by an internal procedure) at a later stage.

Log ID	NGAMS_ER_PROB_STAGING_AREA
Log Number	4016
Log Type	ERROR
Log Text	Problem encountered, while storing data in Staging Area File: %s. Exception: %s
Description	A problem occurred while storing the data on the HTTP connection into the Staging Area on the target disk. This might be caused by that the Staging Area directory is inaccessible, or the target disk as such, or that that there is no more space in the Staging Area.

Log ID	NGAMS_ER_PROB_BACKLOG_BUF
Log Number	4017
Log Type	ERROR
Log Text	Problem encountered, while storing data of file with URI: %s, in Back-Log Buffer: %s. Exception: %s
Description	A problem occurred while storing data in the Back-Log Buffer Area. This might be caused by that the Back-Log Buffer directory is inaccessible, or the target disk as such, or that that there is no more space in the Staging Area.

Log ID	NGAMS_NOTICE_FILE_REINGESTED
Log Number	4018
Log Type	NOTICE
Log Text	Note: File issued with URI: %s was already archived and has been re-ingested!
Description	The file issued was already archived and has been re-ingested into NGAS. This means in practice that an additional copy of the file is available in the NGAS data repository.

Log ID	NGAMS_INFO_ARCHIVING_FILE
Log Number	4019
Log Type	INFO
Log Text	Archiving file with URI: %s
Description	NG/AMS is about to handle/archive the file referenced in the log message. An additional message (additional messages) will follow this one indicating if the archiving was successful or not.

Log ID	NGAMS_INFO_FILE_ARCHIVED
Log Number	4020
Log Type	INFO
Log Text	Successfully archived file with URI: %s
Description	The file referenced by its URI in the log message was successfully handled/archived by NG/AMS

Log ID	NGAMS_ER_MIS_PAR
Log Number	4021
Log Type	ERROR
Log Text	Missing parameter: %s in connection with command: %s.
Description	Together with the command issued, the parameter listed above should have been issued. Since this is not the case, the request cannot be handled.

Log ID	NGAMS_INFO_FILE_AVAIL
Log Number	4022
Log Type	INFO
Log Text	File with File ID: %s is available on NGAS Host with Host ID: %s.
Description	The file with the given File ID, is available and accessible on the NGAS Node contacted.

Log ID	NGAMS_INFO_FILE_NOT_AVAIL
Log Number	4023
Log Type	INFO
Log Text	File with File ID: %s is not available on NGAS Host with Host ID: %s.
Description	The file with the given File ID, is not available/accessible on the NGAS Node contacted.

Log ID	NGAMS_INFO_RETRIEVE_REDIRECT
Log Number	4024
Log Type	INFO
Log Text	Redirection URL: %s
Description	The file requested is not available on the NGAS Node contacted, and the NG/AMS Server on this system cannot act as proxy to query the file and send it back. The file in question can be found at the URL given in the message.

Log ID	NGAMS_INFO_SERVICE_UNAVAIL
Log Number	4025

Log Type	INFO
Log Text	The service: %s is not available on this system.
Description	The service needed for fulfilling the request is not available on this NG/AMS System. Could e.g. be File Retrieval.

Log ID	NGAMS_ER_BAD_FILE
Log Number	4027
Log Type	ERROR
Log Text	Error occurred handling file: %s in DAPI: %s, Error: %s
Description	A DAPI detected a problem/inconsistency in the file being handled. The file is considered as bad and cannot be archived properly.

Log ID	NGAMS_ER_ILL_DPPI
Log Number	4028
Log Type	ERROR
Log Text	Illegal DPPI: %s specified in connection with Retrieve Request. Given DPPI is not defined in the NG/AMS Configuration.
Description	An NG/AMS installation only accepts to execute DPPIs, which are explicitly defined in the NG/AMS Configuration. If a DPPI is specified, which is not defined in the configuration this will be rejected even if this DPPI might be available as such on the NGAS Host handling the request. The problem can be solved by adding the name of the DPPI in the configuration file provided that this DPPI in fact is available on the system and that it is desirable to make this DPPI available for external users.

Log ID	NGAMS_ER_CMD_SYNTAX
Log Number	4028
Log Type	ERROR
Log Text	The combination of parameters given together with the command: %s is illegal. Parameter(s): %s.
Description	The combinations of parameters given together with the command referred to in the error message is illegal (syntactic or semantically) wrong. Check the NG/AMS User's Manual or the online NG/AMS help for further information about the command.

Log ID	NGAMS_ER_CMD_EXEC
Log Number	4029
Log Type	ERROR
Log Text	Problem(s) encountered executing command: %s. Error: %s
Description	A problem or problems was/were encountered while executing the command indicated in the error message. This may be due to an illegal combination of parameters, or due to that prerequisites for executing the command are not fulfilled.

Log ID	NGAMS_ER_DEL_FILE_DISK
Log Number	4030
Log Type	ERROR
Log Text	Problem(s) encountered deleting file from disk. Disk ID: %s, File ID: %s, File Version: %d. Exception: %s.
Description	A problem or problems was/were encountered while trying to delete the referenced file from an NGAS

	disk. Could be due to e.g. that the disk is read-only mounted or that the file itself is read-only.

Log ID	NGAMS_ER_DEL_FILE_DB
Log Number	4031
Log Type	ERROR
Log Text	Problem(s) encountered deleting file info from DB. Disk ID: %s, File ID: %s, File Version: %d. Exception: %s.
Description	A problem or problems was/were encountered while trying to delete the information for the referenced file from an NGAS DB. Could be due to that the DB user has not permission to carry out DELETE SQL statements.

Log ID	NGAMS_INFO_DEL_FILE
Log Number	4032
Log Type	INFO
Log Text	Successfully deleted info for file. Disk ID: %s, File ID: %s, File Version: %d.
Description	The information for a file was successfully removed from the NGAS DB and from the NGAS disk hosting the file.

Log ID	NGAMS_INFO_FILE_DEL_STAT
Log Number	4033
Log Type	INFO
Log Text	File deletion status. Files Selected: %d, Files Deleted: %d, Failed File Deletions: %d.
Description	Status over a REMFILE command indicating 1) How many files were selected for deletion, 2) How many files were deleted, 3) How many attempts to delete file that failed.

Log ID	NGAMS_ER_DEL_DISK
Log Number	4034
Log Type	ERROR
Log Text	Problem encountered deleting files on disk. Disk ID: %s. Exception: %s.
Description	A problem was encountered while trying to delete files on the disk with the Disk ID given in the error message. Maybe the disk is mounted read-only.

Log ID	NGAMS_ER_DEL_DISK_DB
Log Number	4035
Log Type	ERROR
Log Text	Problem encountered deleting disk info from DB. Disk ID: %s. Exception: %s.
Description	A problem was encountered while trying to delete the information for a disk from the NGAS DB. Maybe the DB user used, does not have permission to execute DELETE SQL statements.

Log ID	NGAMS_INFO_DEL_DISK
Log Number	4036
Log Type	INFO
Log Text	Successfully deleted info for disk. Disk ID: %s.
Description	The information for a disk was successfully removed from the NGAS DB and the files on the disk

	successfully deleted.

Log ID	NGAMS_INFO_DEL_DISK_SEL
Log Number	4037
Log Type	INFO
Log Text	Selected info for disk for deletion. Disk ID: %s.
Description	The REMDISK command given, would delete the disk referenced to in the log message if execute=1 would be specified.

Log ID	NGAMS_WA_DEL_DISK2
Log Number	4038
Log Type	WARNING
Log Text	No disk found on NGAS Host: %s with Disk ID: %s. No disk selected.
Description	The REMDISK command given, did not match any disk installed in the contacted NGAS Host.

Log ID	NGAMS_WA_FILE_COPIES
Log Number	4039
Log Type	WARNING
Log Text	One or more files requested to be deleted are not available in at least 3 independent copies within this NGAS system.
Description	For security reasons it is enforced by NG/AMS to have at least two independent copies of each file (defined by a File ID + File Version). In case of a REMDISK or REMFILE command, it will be checked for each file that would be deleted by the request, if it is available within the NGAS system in at least 3 independent copies. An independent copies is defined as instances of one file stored on different storage media.

Log ID	NGAMS_ER_UNKNOWN_MIME_TYPE2
Log Number	4040
Log Type	ERROR
Log Text	Illegal mime-type: %s issued in Archive Request for file with URI: %s. Rejecting request.
Description	The mime-type specified in the Archive Request for the file with the given URI is unknown to this NG/AMS installation. Check the configuration.

Table 26: NG/AMS log definition (FILE: “ngams/ngamsData/ngamsLogDef.xml”).

26 NG/AMS License Conditions

The license conditions provided for NG/AMS is based on the BSD License; more information about this can be obtained following the link: http://opensource.org/licenses/bsd-license.php.

The NG/AMS License text can be obtained either directly in the "ngams" module as "ngams/LICENSE". It can also be obtained by invoking the NG/AMS Server with the parameter "-license": "ngamsServer –license". Likewise it can be obtained using the same parameter for the NG/AMS C- and Python-Clients, i.e.: "ngamsCClient –license" and "ngamsPClient –license". Finally, it can be obtained from the C-API using the function "ngamsLicense()" and from the Python-API using the function "ngamsLicense()".

In the following the content of the NG/AMS License Condition is listed:

```
#     #  #####        #    #    #     #  #####
##    # #     #      #    # #   ##   ## #     #
# #   # #           #    #   #  # # # # #
#  #  # #  ####    #    #     # #  #  #  #####
#   # # #     #   #     ####### #     #       #
#    ## #     #  #      #     # #     # #     #
#     #  #####  #       #     # #     #  #####


                *** LICENSE CONDITIONS ***


Copyright (c) 2001-2002, European Southern Observatory.
All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

        Redistributions of source code must retain the above
        copyright notice, this list of conditions and the
        following disclaimer.

        Redistributions in binary form must reproduce the above
        copyright notice, this list of conditions and the
        following disclaimer in the documentation and/or other
        materials provided with the distribution.

        Neither the name of the European Southern Observatory
        nor the names of its contributors may be used to endorse
        or promote products derived from this software without
        specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.


For further information and in case of questions, please
contact:
                                ngast@eso.org
```

Figure 63: The NG/AMS License Conditions (FILE: "ngams/LICENSE").

27 NG/AMS Commands

This chapter contains a detailed reference to the commands supported by the NG/AMS Server. All the commands are listed and explained, and the command parameters in connection with these are described.

Using the NG/AMS APIs (Chapters 9 and 10) or the NG/AMS Python or C based command line utilities (Section 5.2), the user is assisted in applying the proper parameters. It is recommended to use these when communicating with the NG/AMS Server.

27.1 ARCHIVE Command - Archive Data Files

The ARCHIVE command is used to archive data files. The ARCHIVE command accepts the following parameters:

Parameter	*Mandatory*	*Description*
filename=<file URI>	Yes	The parameter is used to specify the location of the file. In case of an Archive Push Request, NG/AMS may use the given URI, to determine the mime-type of the file. In addition, the temporary filename in the Staging Area is based on the filename (without the path) given in the URI. For an Archive Push Request, the URI is the location (URL) where the file can be picked up by the NG/AMS Server. The location must then be accessible from the NG/AMS Server either via HTTP (http://…), FTP (ftp://…) or directly as file (file://…). Also in this case the mime-type pf the data may be determined from the path if not specified directly by means of the "mime_type" parameter.
mime_type=<mime-type>	No	If the File URI of an Archive Request does not reveal the mime-type of the file to be archived, the mime-type should be specified in the Archive Request. This makes the handling of the request more efficient. An example using this parameter is given in Example 3 below.
no_versioning	No	Used to switch the automatic versioning on/off. If File Versioning is on, a file archived with a File ID already registered in the NGAS DB, will get a new version number (previous number + 1).
wait=0|1	No	With this parameter it is possible to specify if the NG/AMS Server should send back an immediate reply (wait=0) when handling an Archive Request, or if a reply should be sent after the request has been handled (wait=1). In the former case, the client will not know if the Archive Request was handled successfully. The default behavior of the server is to send the *reply* after the Archive Request has been handled.

Table 27: Parameters for the ARCHIVE command.

Example 1: Archiving using Archive Push Technique:
An example of an Archive Push Request can be found in Section 7.1/Figure 20.

Example 2: Archiving using Archive Pull Technique:
The URL for the NG/AMS Server could be something like this:

http://hostx:7878/ARCHIVE?filename=ftp://hosty/data/2002-02-11/Fits1.fits[10]

In this case the NG/AMS Server will pick up the file from the location given. I.e., the client need not to issue the data in the HTTP request. In the example shown, the NG/AMS Server will generate a reply after having handled the Archive Request.

Example 3: NGAS Node to NGAS Node Archiving:
As a small 'curiosity', this example shows an Archive Pull Request, whereby the file URI specified is referring to a file located on another NGAS Node:

http://ngas1:7878/ARCHIVE?mime_type=application/fits&file_uri=http://ngas2:7878/RETRIEVE?file_id=XYZ-2002-02-01T02:23:41.342

In this example, the NG/AMS Server handling the Archive Request will pick up the file from the remote NGAS Host using the file URI which in this case is a Retrieve Request, and will archive it.

[10] The HTTP query string must be encoded according to the specification of the HTTP protocol. Here they are shown un-encoded.

27.2 CLONE Command – Copy Files

The CLONE command is used to create copies of single files or sets of files. In order for the CLONE command to be accepted by an NG/AMS Server, the system must be configured to accept Archive Requests. Also, enough free disk space must be available in the NGAS Host handling the request. If the files to be cloned are located on other NGAS Hosts, these will be requested automatically during the cloning (if possible).

The files to be cloned are selected, based on the parameters File ID, Disk ID and File Version. The interpretations of the various combinations of these parameters are explained in Table 28.

Disk ID	*File ID*	*File Version*	*Interpretation*
Unspecified	**Specified**	Unspecified	Clone one file with the given ID. Latest version of the file is taken.
Specified	**Specified**	Unspecified	Clone one file stored on the given disk. Latest version on that disk is taken.
Unspecified	**Specified**	**Specified**	Clone all files found with the given File Version. Storage location (Disk ID) is not taken into account.
Specified	**Specified**	**Specified**	Clone one file on the given disk with the given File Version.
Specified	Unspecified	Unspecified	Clone all files from the disk with the given ID.
Specified	Unspecified	**Specified**	Clone all files with the given File Version from the disk with the ID given.
Unspecified	Unspecified	**Specified**	Illegal. Not accepted to clone arbitrarily files given by only the File Version.
Unspecified	Unspecified	Unspecified	Illegal. No files specified.

Table 28: Rules applied when selecting files for cloning.

The only way in the present version to abort a Clone Request in progress, is to send an “OFFLINE –force” to the server (a “CLONE –abort” might be provided at a later stage). Note, in this version of NG/AMS it is not possible to specify a target disk for the cloning. This will be provided at a later stage.

The parameters for the CLONE command are:

Parameter	*Mandatory*	*Description*
file_id=<file ID>	See Table 28	The ID of the files to consider.
file_version=<file version>	See Table 28	The File Version of the files to consider.
disk_id=<disk ID>	See Table 28	The Disk ID of the files to consider.

Table 29: Parameters for the CLONE command.

As a result of the CLONE command, a File Cloning Status Report is send back indicating which files were cloned and the target names of these.

27.3 EXIT Command - Terminate Server

The EXIT command is used to make the NG/AMS Server exit. The EXIT command does not accept any parameters.

27.4 INIT Command - Re-Initialize the System

The INIT command is used make the NG/AMS Server re-initialize. This means that it will first go Offline, load the configuration and subsequently go Online. The INIT command does not accept any parameters.

27.5 LABEL Command - Generating Disk Labels

The LABEL command is used to print out labels to be put on the disk cases. The label is the Logical Name of a disk. The LABEL command accepts the following parameters:

Parameter	*Mandatory*	*Description*
slot_id=<slot ID>	Yes	The ID of the slot in which the disk is installed.
host_id=<host ID>	No	The host in which the disk is installed. If this is not specified, the local host is assumed.

Table 30: Parameters for the LABEL command.

An example of a label generated by NG/AMS (by means of the Label Plug-In), can be found in Section 3.10.

27.6 OFFLINE Command - Bring System to Offline State

The OFFLINE command is used to make the NG/AMS Server go Offline. The OFFLINE command accepts the following parameter:

Parameter	*Mandatory*	*Description*
force	No	Force the system to Offline State even though an action is in progress like file cloning.

Table 31: Parameters for the OFFLINE command.

Usage of the "force" option should be done with great care, as operations may be interrupted before termination, leaving the system in an 'undefined' condition.

27.7 ONLINE Command - Bring System to Online State

The ONLINE command is used to make the NG/AMS Server go Online. The ONLINE command does not accept any parameters.

27.8 REGISTER Command - Register Existing Files on a Disk

The REGISTER command is used to register files already stored on an (NGAS) disk. It is possible to register single files, or entire sets of files by specifying a starting path from which NG/AMS will look for files. Only files that are known to NG/AMS (with a mime-type defined in the configuration), will be taking into account. It is also possible to explicitly specify a comma separated list of mime-types that will be registered. Files with other mime-types than specified in this list will be ignored.

Parameter	*Mandatory*	*Description*
mime_type	No	Comma separated list of mime-types to take into account. A single mime-type can also be specified.
path	Yes	The starting path under which NG/AMS will look for candidate files to register. It is also possible to specify a complete path, whereby only a single file will be registered.

Table 32: Parameters for the REGISTER command.

As a response to the REGISTER command, and Registration Report is generated, indicating which files where registered.

27.9 REMDISK Command – Remove Information about Disks

The REMDISK command is used to remove information about entire disks from NGAS. *Great caution should therefore be applied when using this command!* Both the information about the Storage Media and the files stored on, will be removed. NG/AMS will not accept to remove a file from the system unless there are at least three (3) independent copies of the file. Three independent copies refers to three copies of the file stored on three independent Storage Media. In order for the REMDISK command to be accepted the system must be configured to allow remove requests (CFG: "NgamsCfg.Ngams:AllowRemoveReq"). If the command is executed without the "execute" parameter, the information about the disk is not deleted, but a report is generated indicating what will be deleted if the execution is requested (execute=1).

The REMDISK command takes the following input parameters:

Parameter	*Mandatory*	*Description*
disk_id=<disk ID>	Yes	Disk ID for the disk to remove.
execute=0|1	No	If execute is not specified or specified as 0, no information will be deleted, but a report will be send back to indicate what will be deleted if the command is executed. If execute is specified as 1, the information in the DB and on the disk itself is deleted.

Table 33: Parameters for the REMDISK command.

As a result of the REMDISK command, a report is send indicating which disk that was removed from the system.

27.10 REMFILE Command – Remove Files from the System

The REMFILE command is used to remove information about files and the files themselves from NGAS. *Great caution should therefore be applied when using this command!* NG/AMS will not accept to remove a file from the system unless there are at least three (3) independent copies of the file. Three independent copies refers to three copies of the file stored on three independent storage media. In order for the REMFILE command to be accepted the system must be configured to allow remove requests (CFG: "NgamsCfg.Ngams:AllowRemoveReq"). If the command is executed without the "execute" parameter, the information about the file(s) is not deleted, but a report is generated indicating what will be deleted if the execution is requested (execute=1).

The selection of the files to be scheduled for deletion is done based on the parameters Disk ID, File ID and File Version. The rules for this are shown in Table 34.

Disk ID	*File ID*	*File Version*	*Interpretation*
Specified	Unspecified	Unspecified	No files selected.
Specified	**Specified**	Unspecified	All files with the given File ID pattern on the disk with the given ID will be selected. No specific File Version will be taken into account.
Specified	Unspecified	**Specified**	No files are selected.
Specified	**Specified**	**Specified**	The referenced file(s) with the given File ID and File Version on the given ID is selected (if this exists).
Unspecified	**Specified**	Unspecified	All files matching the given File ID pattern on the contacted NGAS Host are selected.
Unspecified	**Specified**	**Specified**	All files with the given File ID pattern and the given File Version are selected without taking the Disk ID into account.
Unspecified	Unspecified	**Specified**	No files are selected.
Unspecified	Unspecified	Unspecified	No files are selected.

Table 34: Selection rules applied for the REMFILE command.

The REMFILE command takes the following input parameters:

Parameter	*Mandatory*	*Description*
disk_id=<disk ID>	See Table 34	Disk ID for the disk to remove.
execute=0|1	No	If execute is not specified or specified as 0, no information will be deleted, but a report will be send back to indicate what will be deleted if the command is executed. If execute is specified as 1, the information in the DB and on the disk itself is deleted.
file_id=<file ID>	See Table 34	ID of files to take into account.
file_version=<file version>	See Table 34	Version of files to take into account.

Table 35: Parameters for the REMFILE command.

As a result of the REMFILE command, a report is send back, indicating which disk that were moved from the system, or alternatively, if the "execute" is 0 or unspecified, a list of files that will be deleted if the command is executed is returned.

27.11 RETRIEVE Command - Retrieve & Process Files

The RETRIEVE command is used to retrieve archived data files from an NGAS Node. The RETRIEVE command accepts the following parameters:

Parameter	*Mandatory*	*Description*
file_id=<file ID>	Yes	ID of file to retrieve.
file_version=<file version>	No	Version of the file to retrieve.
internal=<filename>	No	Retrieve the contents of an internal file. Could e.g. be the a source file or the log definition file. This is mostly intended for maintenance/trouble-shooting purposes.
ng_log	No	Retrieve the contents of the NG/AMS Local Log File (see 3.3).
processing_pars=<DPPI>	No	With this parameter it is possible to specify a DPPI, which is invoked to process the data before sending it back. NG/AMS will send back the result of the processing, and not the original file.

Table 36: Parameters for the RETRIEVE command.

It is possible to receive an HTTP redirection response as response to the Retrieve Request. In this case the client must re-send the Retrieve Request to the alternative URL given in the redirection response. See also Section 7.3.

27.12 STATUS Command - Query System Status & Other Information

The STATUS command is used to query various status information from the NG/AMS Server. The STATUS command accepts the following parameters:

Parameter	*Mandatory*	*Description*
<no parameters>	No	In this case a reply is returned which contains an NG/AMS Status document. An example of such a status document can be found in Section 3.8.
disk_id=<disk ID>	No	Query information about a disk referred to by its Disk ID. The reply is an NG/AMS Disk Status XML document. An example of this can be found in Section 22.2.
file_id=<file ID>	No	Query information about a file with a given File ID. The reply is an NG/AMS XML Status document as shown in Section 22.3.
file_version=<file version>	No	Used to specify specific version of file to query information for.
configuration_file	No	Query the configuration used by an NG/AMS server. The result is a complete NG/AMS Configuration XML document as shown in Section 6.3.
file_access=<file ID>	No	This parameter is used to make an NG/AMS Server probe if it can physically access a file. The body of this request must contain an NG/AMS File Status XML document with the detailed information about the file. It is therefore necessary to issue also the "content-length" and "content-type" HTTP headers followed by the file status in the request. This command is thus similar to an Archive Push request, refer to Section 7.1 for further information about this issue.
flush_log	No	Used to make the NG/AMS Server flush the logs it may have cached internally, into the Local Log File if such is specified.

Table 37: Parameters for the STATUS command.

It is only possible to specify one of the parameters at a time.

27.13 SUBSCRIBE Command – Subscribe to Data from NGAS Host

The SUBCRIBE command is used by a Data Subscriber to subscribe to a certain kind of data becoming available (and which is available at the time of the subscription). The issue of Data Subscription is described in detail in Section 4.2. The parameters of the SUBSCRIBE command are listed in Table 38.

Parameter	*Mandatory*	*Description*
filter_plug_in=<plug-in>	No	Name of a Filter Plug-In (see Chapter 21) to invoke on the data to determine whether to deliver this to the client or not.
plug_in_pars=<pars>	No	A set of parameters to transfer to the Filter Plug-In when it is invoked.
priority=<prio>	No	Priority for delivering data to this Data Subscriber. The lower the number, the higher the priority. Clients with a higher priority, get more CPU time in connection with the data delivery.
start_date=<ISO8601>	No	Date from which the data to deliver is taken into account. If not specified the time when the SUBSCRIBE command was received is taken as start date.
url=<delivery URL>	Yes	The URL to which the data will be delivered. On the client side a corresponding HTTP server must be ready to receive requests (data) via the given URL.

Table 38: Parameters for the SUBSCRIBE command.

Note, a Data Subscriber that has subscribed remains subscribed also if the HTTP server, which handles the receiving of data on the client side terminates. NG/AMS will back-log data for that client, and when it re-subscribes, the back-logged data will be delivered. If it not desirable, the client should un-subscribe it-self by submitting an UNSUBSCRIBE command.

27.14 UNSUBSCRIBE Command – Unsubscribe a Previous Data Subscription

Used by Data Subscribers to un-subscribe a previously established subscription for data. If NG/AMS holds back-logged buffered subscription data for the client from a previous subscription, the Subscription Back-Log will be reset. The parameter of the UNSUBSCRIBE command is listed in Table 39

Parameter	*Mandatory*	*Description*
url=<delivery URL>	Yes	The URL, which was submitted when the client subscribed itself. The Delivery URL is used by NG/AMS to identify each Data Subscriber.

Table 39: Parameters for the UNSUBSCRIBE command

If the client was not subscribed, the UNSUBSCRIBE command has no effect.

28 Index

M

N

O

P

Q

R

S

T

U

V

W

X